2019-01-01

Combination Of Resampling Based Lasso Feature Selection And Ensembles Of Regularized Regression Models

Abhijeet R. Patil
University of Texas at El Paso

COMBINATION OF RESAMPLING BASED LASSO FEATURE SELECTION AND ENSEMBLES OF REGULARIZED REGRESSION MODELS

ABHIJEET R PATIL

Master's Program in Computational Science

APPROVED:

Sangjin Kim, Ph.D., Chair

Ming-Ying Leung, Ph.D., Co-Chair

Sourav Roy, Ph.D.

Lin Li, Ph.D.

Stephen Crites, Ph.D.
Dean of the Graduate School

to my

MOTHER and FATHER

with love

COMBINATION OF RESAMPLING BASED LASSO FEATURE SELECTION AND ENSEMBLES OF REGULARIZED REGRESSION MODELS

by

ABHIJEET R PATIL, B.E., M.Tech

THESIS

Presented to the Faculty of the Graduate School of

The University of Texas at El Paso

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE

Computational Science Program

THE UNIVERSITY OF TEXAS AT EL PASO

December 2019

Acknowledgements

I would like to express my most sincere gratitude to my advisor, Dr. Sangjin Kim for his wise guidance, constant encouragement, and advice throughout my research. I sincerely appreciate his generous time and ideas. I would like to thank my co-advisor, Dr. Ming-Ying Leung for her constant support, supervision in my study and thesis. I am indebted and thankful to both of my advisors. I have greatly benefited from their expertise. I could never reach to this point without enormous help from them.

I would also like to thank my committee members; Dr. Sourav Roy of Biological Sciences Department and Dr. Lin Li of Physics Department, all at The University of Texas at El Paso. Their encouragement, insightful comments, additional guidance were invaluable to my research. I am also thankful to Dr. Jonathon E Mohl for his valuable suggestions and support.

I will forever be thankful to my undergraduate mentor, Prof. A.M. Padma Reddy. He has always been helpful in providing advice many times during my engineering. I still think fondly of my time as an undergraduate student in his classes. His enthusiasm and love for teaching is contagious. Padma Reddy sir is the reason i decided to go to pursue a career in teaching and research.

I especially thank my father Revansiddappa.S.Patil, my mother Jyothi.R.Patil, brother Shankar.R.Patil, sister Preeti.R.Patil, and my wife Dikshita.A.Patil for their constant inspiration, loving support and encouragement in all my decisions. I undoubtedly could not have done this without you all.

Abstract

In high-dimensional data, the performance of various classifiers is largely dependent on the selection of important features. Most of the individual classifiers using existing feature selection (FS) methods do not perform well for highly correlated data. Obtaining important features using the FS method and selecting the best performing classifier is a challenging task in high throughput data. In this research, we propose a combination of resampling based least absolute shrinkage and selection operator (LASSO) feature selection (RLFS) and ensembles of regularized regression models (ERRM) capable of handling data with the high correlation structures. The ERRM boosts the prediction accuracy with the top-ranked features obtained from RLFS. The RLFS utilizes the LASSO penalty with sure independence screening condition to select the top k ranked features. The ERRM includes five individual penalty based methods: LASSO, adaptive LASSO (ALASSO), elastic net (ENET), smoothly clipped absolute deviations (SCAD), and minimax concave penalty (MCP). It is built on the idea of bagging and rank aggregation. Upon performing simulation studies and applying to smokers cancer gene expression data, we demonstrated that the proposed combination of ERRM with RLFS achieved superior performance in accuracy and geometric mean.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

With the advances of high throughput technology in biomedical research, large volumes of high-dimensional data are being generated [1, 2, 3]. Some of the examples of such data can be found in microarray gene expression [4, 5, 6] data, RNA-seq [7], genome-wide association study (GWAS) [8, 9] data, and DNA-methylation [10, 11]. These data are high dimensional in nature, where the total count of features is significantly larger than the number of samples ($p >> n$) and termed as the curse of dimensionality. Although this is one of the major problems, there are many other problems such as noise, redundancy and over parameterization. To deal with these problems, many two-stage approaches of FS and classification algorithms have been proposed in machine learning over the last decade. While the FS methods are used to reduce the dimensionality of data by removing noisy and redundant features that help in selecting the truly important features, the classification algorithms help increase the prediction performance.

The FS methods are classified into rank-based and subset methods [12, 13]. Rank-based methods rank all the features with respect to their importance based on some criteria. Although there is a lack of threshold to select the optimal number of top-ranked features, this can be solved using Sure Independence screening (SIS) [14] condition. Some of the popular rank-based FS methods used in bioinformatics are Information Gain [15], Chi-square [16] and Minimum Redundancy Maximum Relevance [17]. Subset methods [18] are the ones where the subset of features are selected with some pre-determined threshold based on some criteria but these methods need more computational time in high-dimensional data setting and lead to an NP-hard problem [19]. Some of the popular subset methods include Boruta [20], Fisher score [21] and Relief [22].

For the classification of gene expression data, there are non-parametric based popular algorithms used, such as Random Forests [23], Adaboost [24], and Support Vector Machines [25]. While the Random Forests and Adaboost are based on the concept of decision trees, the Support Vector Machines is based on the idea of hyperplanes. In addition to the above, there are parametric machine learning algorithms such as penalized logistic regression (PLR) models that have five different penalties which are predominantly popular in high-dimensional data. The first two classifiers are Lasso [26] and Ridge [27] that are based on L1 and L2 penalties. The third classifier is a combination of these and is termed as elastic net [28]. The other two PLR classifiers are SCAD [29] and MCP [30] which are based on non-concave and concave respectively. All these individual classifiers are very common in machine learning and bioinformatics [31]. However, in highly correlated gene expression data, these individual classifiers do not perform well in terms of prediction accuracy. To overcome the issue of individual classifiers, ensembles classifiers are proposed [32, 33]. The ensemble classifiers are bagging and aggregating methods [34, 35] that are employed to improve the accuracy of several "weak" classifiers [36]. The tree-based method Classification by ensembles from random partitions (CERP) [37] showed good performance but is computer-intensive. The ensembles of logistic regression models (LORENS) [38] for high-dimensional data were proven to be good for classification. However, they decrease in performance when there are small number of true important variables in the high-dimensional space because of random partition.

In this research, we introduce the proposed combination of FS and classifier, the resampling based Lasso Feature Selection (RLFS) method for ranking the features and Ensembles of Regularized Regression Models (ERRM) for classification purposes. The resampling approach is proven to be one of the best FS screening step in a high-dimensional data setting [13]. The RLFS uses the selection probability with lasso penalty and the threshold for selecting the top-ranked features is set using k-SIS condition and these selected features are applied to the ERRM to achieve the best prediction accuracy. The ERRM uses five individual regularization models, Lasso, Adaptive lasso (Alasso), Elastic Net (ENET), SCAD, and

MCP. These methods are used in each individual tree with the bootstrapped samples from the training data in the learning phase. This step is repeated for t times. The combination of bagging and weighted rank aggregation is employed along with majority voting to select the best performing classification method M_i given the bootstrap sample B_i.

The rest of the thesis is organized as follows. In Chapter 3, the RLFS and ERRM along with the other FS and classification methods are explained in detail. In Chapter 4 we show that the proposed combined framework of RLFS and ERRM has superior performance in comparison to other combinations of existing FS methods and individual classifiers. In Chapter 4.1 we show the results with explanation of our extensive simulation studies and in Chapter 4.2 we show the performance of our proposed methods outperforming all other methods on the gene expression smokers cancer (SMK-CAN-187) data. We discuss our findings briefly in the Chapter 5 and finally Chapter 6 concludes the thesis.

Chapter 2

Literature Review

This chapter describes the various feature selection and classification algorithms used in high-dimensional data. The first section explains the different types of feature selection and the computational algorithms involved. In the second section, we will discuss the different classification algorithms used in machine learning.

2.1 Feature Selection

There are many challenges faced when we deal with high-dimensional data, such as high-performance computing, overfitting, and redundant features. To overcome such problems, in high-dimensional data, two-stage approaches of filtering and classification was proposed [13]. Feature selection (FS) is a process of removing the noisy and redundant features from the data. It helps in boosting the performance of a classification algorithm not just in terms of accuracy but also in computational time.

There Feature selection can be divided into three different categories namely filter-based, embedded, and wrapper methods.

2.1.1 Filter-based Methods

The filter-based approaches are independent of the classification methods; therefore, they are computationally faster than the wrapper and embedded methods. The relevant features are selected based on distances, entropy, and uncertainty. There are many algorithms developed. Some of the best examples include Relief [22], which uses the distance-based metric function. The ReliefF [39], which is a modified version of Relief, is developed to han-

dle the multi-class problems. The minimum redundancy maximum relevance (MRMR) [17] and mutual information-based feature selection method (MIFS) [40] are the FS methods that rely on mutual information criteria. The mutual information is calculated between the individual feature and response label. The FS method conditional mutual information maximization (CMIM) [41] recursively chooses the features that provide the maximum mutual information with the response class. The various entropy-based methods are developed, Information gain [15], gain ratio, and symmetrical uncertainty.

2.1.2 Embedded-based Methods

The embedded-based methods incorporate the FS process inside the classification method, which helps in performing the gene selection and classification simultaneously. It helps reduce the computational time than the wrapper method; however, it is expensive when compared to filter-based methods. The embedded method includes the pruning method, built-in mechanism, and regularization methods. In the pruning method, all the features are considered during the training phase for building the classification model. The features with lower correlation coefficient values are removed recursively using the support vector machines. In the built-in FS method, the important features are ranked based on their importance. The variable importance (varImp) measure in the random forest method is the best example of a built-in measure. In the regularized methods, the penalized regression models are based on penalties and are popular in high-dimensional data for variable selection and classification purposes. Some of the examples of the sparse variable selection models include lasso, adaptive lasso, elastic net, SCAD, and MCP. These methods are discussed in Chapter 3.

2.1.3 Wrapper-based Methods

Wrapper-based selects the feature subsets using classification algorithm and searching techniques. The examples of the former approach include forward selection, backward elimi-

nation, and recursive feature elimination. The latter approach includes hill climbing and best-first search strategies using the decision trees and naive Bayes classifiers. The wrapper methods are computationally expensive because the features are selected based on the performance of the classifiers or the searching techniques, which is a recursive process.

2.2 Classification Methods

Depending on the type of data, the classification algorithms are classified into two categories, supervised classification and unsupervised clustering.

2.2.1 Supervised Classification

There is a broad range of algorithms that can be applied for labeled data such as tree-based methods, discriminant analysis, and penalized regression models. The popular tree-based methods include random forest and adaptive boosting. The random forests are built on the concept of decision trees. The idea is to operate as an ensemble method instead of relying on a single method. It is based on the concept of bagging and majority voting. The Adaptive boosting method is an ensemble learning technique where it improves the single weak boosting algorithm through an iterative process. The support vector machines detect the maximum margin hyperplane by maximizing the distance between the hyperplane and the closest dot. The maximum margin indicates that the classes are well separable and correctly classified. These methods are discussed in Chapter 3.

2.2.2 Unsupervised Clustering

In this section, the various unsupervised machine learning methods used for unlabeled data are discussed. Some of the examples include; K-Nearest-Neighbor and K-means clustering. Clustering algorithms group the samples based on some sort of similarity metric that is computed for features. In the context for gene expression data, genes represented as fea-

tures are grouped into classes on the terms of similarity in their expression profiles across tissues, cases or condition [42]. Clustering algorithms divide the samples into a predetermined number of groups in a way that maximizes a specific function. Although there are many such clustering methods for handling the unlabelled data, most of them do not work well in a high-dimensional setting. Therefore reducing the size of data through the various dimensionality reduction methods becomes a necessity. After the reduced data is obtained, the cluster-based models can be applied for prediction purposes. Some of the popular dimensionality reduction techniques include principal component analysis and linear discriminant analysis [3]. These methods reduce the size of data by transforming the original features in high-dimensional space to fewer dimensions.

The KNN classifier is known for its non-parametric (It is a quasi model where there are underlying assumptions made) behavior. It can be used for supervised and unsupervised learning. It is a lazy algorithm, which means that there is no explicit training phase as the decisions are made on entire training data. The downside of this algorithm is that it is computationally expensive in terms of cost and time.

Chapter 3

Materials and Methods

This section is divided mainly into three sections. The first section describes the FS methods, the second section explains the classification algorithms and the third section shows the metrics used for measuring the performance of the models. In section 3.1, the RLFS method is explained followed by a brief description of the other commonly used rank based FS methods in binary classification problems such as Information Gain (IG), Chi-squared (Chi2) and Minimum Redundancy Maximum Relevance (MRMR). In section 3.2, we first explain in detail the ERRM method followed by brief description of the popular classification algorithms in the field of bioinformatics.

The performance of ERRM is compared with the support vector machines, embedded logistic regression models, and tree-based ensemble methods such as random forests and adaptive boosting classifiers. The programs for all the experiments are written using R software [43]. The FS and Classification is performed with the packages [44, 45, 46, 47, 48, 49] obtained from CRAN. The weighted rank aggregation is evaluated with RankAggreg package obtained from [50]. The SMK-CAN-187 data is obtained from [51], some of the applications of the data can be found in the articles [52, 53] where the importance of screening approach in high dimensional data is elaborated.

3.1 Rank Based Feature Selection Methods

With the gain in popularity of high dimensional data in bioinformatics, the challenges to deal with it also grows. In gene expression data, having large p and small n problems, the n represents the samples as patients and p represents the features as genes. Dealing with

such a large number of genes that are generated by conducting large biological experiments involves computationally intensive tasks that become too expensive to handle. The performance drops when such a large number of genes are added to the model. To overcome this problem, employing FS methods becomes a necessity. In statistical machine learning, there are many FS methods developed to deal with the gene expression data. But most of the existing algorithms aren't completely robust applications to the gene expression data. Hence, we propose an FS method that ranks the features based on some criteria explained in the next section. We also explain some other popular FS methods in classification problems such as IG, Chi2, and MRMR.

3.1.1 The Proposed Resampling based Lasso Feature Selection

From [13] we see that the resampling based FS is relatively more efficient in comparison to the other existing FS methods in gene expression data. The RLFS method is based on the lasso penalized regression method and resampling approach employed to obtain the ranked important features using the frequency.

The Least absolute shrinkage and selection operator (Lasso) [26] estimator is based on L1-regularization. The L1-regularization method limits the size of coefficients pushes the unimportant regression coefficients to zero by using the L1 penalty. Due to this property, variable selection is achieved. It plays a key role in achieving better prediction accuracy along with the gene selection in bioinformatics.

The lasso penalty is shown below:

$$\hat{\beta}_{lasso} = \underset{\beta}{\operatorname{argmin}} \left[-\sum_{i=1}^{n} \{ y_i \log(\pi(y_i = 1|x_i)) + (1 - y_i) \log(1 - \pi(y_i = 1|x_i)) \} + \lambda \sum_{i=1}^{p} |\beta_i| \right] \quad (3.1)$$

The selection probability $S_p(f)$ of the features based on the lasso is shown in the below equation.

$$S(f_p) = \frac{1}{R} \sum_{i=1}^{R} \frac{1}{L} \sum_{j=1}^{L} \mathrm{I}(\beta_{ijp} \neq 0), \text{ for } f = 1, 2, ..., p \quad (3.2)$$

The k-SIS criteria to select the top k ranked features is defined by,

$$\left\lceil k \times \frac{n}{log(n)} \right\rceil \tag{3.3}$$

where R is defined by the total number of resampling, L is total number of λ values, p is defined by index of features, n is total number of samples, $\beta_{i,j,p}$ is defined as regression coefficient of pth index of feature f and $I()$ indicator variable. For each of the R number of resampling and L number of values of λ are considered to build the variable selection model. The 10-fold cross validation is considered while building the model.
After ranking the features using the RLFS method, we employ the k-SIS approach to select the top features based on (3.3) where k is set to 2. The number of true important variables selected among the top k-SIS ranked features are calculated in each iteration and the average of this is taken over 100 iterations.

3.1.2 Information Gain

The Information Gain (IG) [15] is simple and one of the widely used FS methods. This univariate FS method is used to assess the quantity of information shared between the feature space X and the response variable Y. It provides an ordered ranking of all the features having a strong correlation with the response variable that helps to obtain good classification performance. The information gain between the i-th feature X_i and the response labels Y are given as follows:

$$\mathrm{IG(X}_i, \mathrm{Y)} = \mathrm{H(X}_i) - \mathrm{H(X}_i|\mathrm{Y}) \tag{3.4}$$

where $H(X_i)$ is entropy of X_i and $H(X_i|Y)$ is entropy of X_i given Y. The entropy [54] of X is defined by the following equation:

$$\mathrm{H(X}_i) = \sum_{x_i \in X} \pi(x_i) log_2(\pi(x_i)) \tag{3.5}$$

where x_i indicates discrete random variable X, $P(x_i)$ gives the probability of x_i on all values of x.

Given the random variable Y, the conditional entropy of X is:

$$\mathrm{H(X}_i\mathrm{|Y)} = \sum_{y_j \in Y} \pi(y_j) \sum_{x_i \in X} \pi(x_i|y_j) log_2(\pi(x_i|y_j)) \tag{3.6}$$

where $\pi(y_i)$ is the prior probability of y_i, $\pi(x_i|y_j)$ is conditional probability of x_i in a given y_j that shows the uncertainty of X given Y.

3.1.3 Chi-square test

Chi-square test (Chi2) [16, 55, 56, 54, 57] is a statistical test of independence used to determine the significant relationship between two categorical variables. The basic rule is that the features having a strong dependency on the class labels are selected and the features independent of the class labels are ignored. Two events X and Y are said to be independent of each other if:

$$\pi(XY) = \pi(X)\pi(Y) \;\; or \;\; \pi(X|Y) = \pi(X) \;\; and \;\; \pi(Y|X) = \pi(Y) \tag{3.7}$$

These events corresponds to the occurrence of feature and the response variable. Based on the following equation, we can rank the terms.

$$\tilde{\chi}^2(f_i,\ y) = \sum_{e_f \in \{0,1\}} \sum_{e_y \in \{0,1\}} \frac{(N_{e_j e_y} - E_{e_j e_y})^2}{E_{e_j e_y}} \tag{3.8}$$

where e_f and e_y are the feature term and the response class respectively, N is the observed frequency, and E is the expected frequency.

3.1.4 Minimum Redundancy Maximum Relevance

The minimum redundancy and maximum relevance method (MRMR) [17] is built on optimization criteria of mutual information (redundancy and relevance) hence it is also defined under mutual information based methods. The MRMR criterion is defined as follows:

$$\mathrm{J_{MRMR}}(X_k) = \mathrm{I}(X_k; Y) - \frac{1}{|S|} \sum_{X_j \in S} \mathrm{I}(\mathrm{X}_k; X_j) \tag{3.9}$$

where X_k represents each feature, Y represent the response variable, and S represents the subspace containing the features. The importance of feature X_k is represented by the value of $J(X_k)$. Higher the value of $J(X_k)$, more important the feature.

3.2 Classification Algorithms

Along with gene selection, improving prediction accuracy when dealing with high-dimensional data has always been a challenging task. There is a wide range of popular classification algorithms used when dealing with high throughput data such as tree-based methods [58], support vector machines, and penalized regression models [59]. These popular models are discussed briefly in this section.

3.2.1 The Proposed Ensembles of Regularized Regression Models

We firstly divided the original data $X_{n\times p}$into 70% of training and 30% of the testing set. The classifiers are fitted on $X_{t\times p}$ and the class labels y as training data set to predict the classification of y using $X_{(n-t)\times p}$ of test set.

The detailed procedure is as follows. The training data $X_{t\times p}$ is given to the proposed RLFS and the new reduced feature set $X_{t\times f}$ is obtained which is used as new training data for the proposed ERRM model. Here, t is the samples included in training data, $n-t$ is the samples included in testing data, p is the total count og features and f is the reduced number of features after FS.

Lasso, Alasso, ENET, SCAD, and MCP are the five individual regularized regression models included as base learners in our ERRM. The role of bootstrapped aggregation or bagging is to reduce the variance through averaging over an "ensemble" of trees which will improve the performance of weak classifiers. $B = B_1^k, ..., B_M^k$ is the number of random

bootstrapped samples obtained from reduced training set $X_{t\times f}$ with corresponding class label y. The fived regularized regression models are trained on each bootstrapped sample B named as sub-training data leading to $5 \times B$ models. These five regularized models are then trained using the 10-fold cross validation to predict the classes on the out of bag samples called sub-testing data where the best model fit in each of the five regularized regression model is obtained. Henceforth, in each of the five regularized model, the best model is selected and the testing data $X_{(n-t)\times p}$ is applied to obtain the final list of predicted classes for each of these models. For binary classification problems, in addition to accuracy, the sensitivity and specificity are largely sought. The J evaluation metrics are computed for each of these best models of five regularized models. In order to get an optimized classifier using all the evaluation measures J is important and this is achieved using weighted rank aggregation. Here, each of the regularized models are ranked based on the performance of J evaluation metrics. The models are ranked based on the increasing order of performance, in case of matching score of accuracy of two or more models, other metrics such as sensitivity and specificity will be considered. The best performing model amongst the five models is obtained based on these ranks. This procedure is repeated to obtain the best performing model in each of the tree T. Finally, majority voting procedure is applied over the T trees to obtain a final list of predicted classes. The test class label is applied to measure the final J measures for assessing the performance of the proposed ensembles. The majority voting is defined as: $\underset{c}{\operatorname{argmax}} \sum_{i=1}^{5} I(\hat{y}_i = c)$.

The complete workflow of the proposed RLFS-ERRM framework is shown in Figure 3.1.

3.2.2 Logistic Regression

Logistic regression (LR) is perhaps one of the basic and popular models used while dealing with binary classification problems [60]. Logistic regression for dealing with more than two classes is called multinomial logistic regression. The primary focus here is on the binary classification. Given the set of inputs, the output is a predicted probability that the given

Algorithm 1 Proposed ERRM

Step 1: Obtain new training data $X_{t\times f}$ with most informative features using the proposed RLFS method.

Step 2: Draw bootstrap samples from $X_{t\times f}$ and apply it to each of the regularized methods to be fitted with 10-fold cross validation.

Step 3: Apply out of bag samples (OOB) not used in bootstrap samples to the above fitted models to choose the best model using J performance metrics.

Step 4: Repeat steps 2 and 3 untill getting 100 bootstrap models.

Step 5: Apply testing set $X_{(n-t)\times f}$ to each of 100 models to aggregate votes of classification.

Step 6: Predict classification of each sample by the rule of majority voting in the testing set.

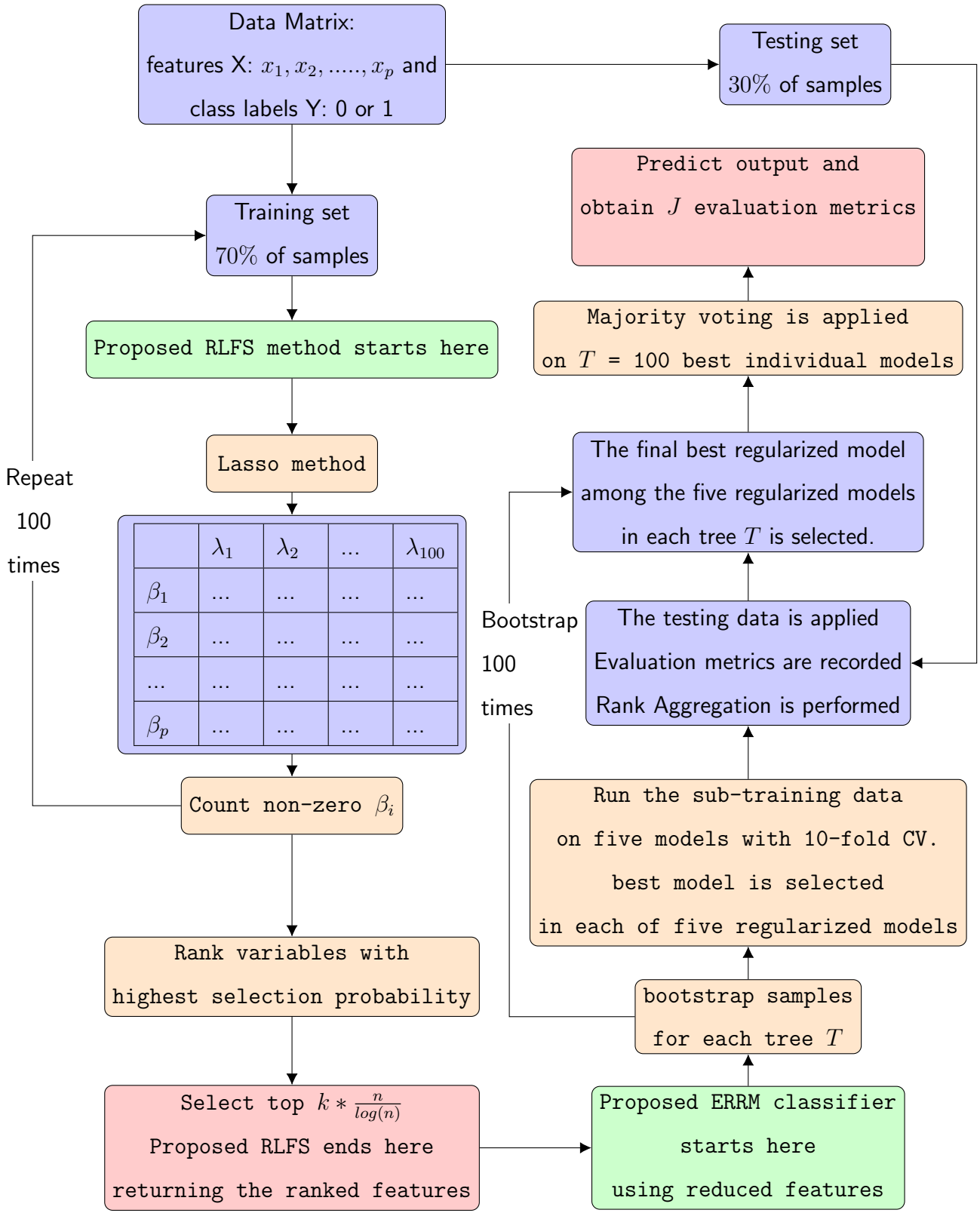

Figure 3.1: The complete workflow depicting the proposed combination of RLFS-ERRM framework

input point belongs to a particular class. The output is always between [0, 1]. Logistic regression is based on the assumption that the original input space can be divided into two separate regions, one for each class, by a plane. This plane helps to discriminate between the dots belonging to different classes and is called as linear discriminant or linear boundary.

One of the limitation is the number of parameters that can be estimated needs to be smaller and should not exceed the number of samples.

3.2.3 Regularized Regression Models

Regularization is a technique used in logistic regression by employing penalties to overcome the limitations of dealing with high-dimensional data. Here, we discuss the PLR models such as Lasso, Adaptive lasso, ENET, SCAD, and MCP. These five methods are included in the proposed ERRM and also tested as independent classifiers for comparing performance with the ERRM.

Let the expression levels of features in i^{th} sample be represented as $X_i = (x_{i1}, x_{i2}, ..., x_{ip})$ for $i = 1, ..., n$ and p is the number of features. The response variables, $y_i \in \{0, 1\}$ where y_i=0 means that i^{th} individual is in the non disease group and y_i=1 is disease group.

The logistic regression equation:

$$\log\left(\frac{\pi(y_i = 1|x_i)}{1 - \pi(y_i = 1|x_i)}\right) = \beta_0 + \beta \mathrm{x} \tag{3.10}$$

where $i = 1....n$ and $\beta = (\beta_1...\beta_p)^T$.

From logistic regression (3.10), the log-likelihood estimator is shown as below:

$$L(\beta_0, \beta) = \sum_{i=1}^{n} \{y_i \log(\pi(y_i = 1|x_i)) + (1 - y_i)\log(1 - \pi(y_i = 1|x_i))\}. \tag{3.11}$$

Logistic regression offers the benefit by simultaneous estimation of the probabilities $\pi(x_i)$ and 1-$\pi(x_i)$ for each class. The criterion for prediction is $I\{\pi(x_i) \geq 0.5\}$, where $I(\cdot)$ is an indicator function. The penalty term is added to the negative log-likelihood function:

$$L_{plr}(\beta_0, \beta) = L(\beta_0, \beta) + p(\beta_i). \tag{3.12}$$

where $p(\beta_i)$ is a penalty function.

The parameters for PLR is estimated by minimizing above function:

$$\hat{\beta}_{plr} = \underset{\beta}{\operatorname{argmin}} \Bigg[- L_{plr}(\beta_0, \beta) \Bigg], \tag{3.13}$$

The lasso penalized regression method is defined in the (3.1). It is a widely used method in variable selection and classification purpose in high dimensional data. It is one of the five methods used in the proposed ERRM for classification purposes.

The oracle property [29] is having consistency in variable selection and asymptotic normality. The lasso works well in subset selection, however, it lacks the oracle property. To overcome this, different weights are assigned to different coefficients and this is termed as weighted lasso called adaptive lasso. The adaptive lasso penalty is shown below:

$$\hat{\beta}_{alasso} = \underset{\beta}{\operatorname{argmin}} \Bigg[- L(\beta_0, \beta) + \lambda \sum_{i=1}^{p} w_i |\beta_i| \Bigg], \tag{3.14}$$

The ridge estimator [27] uses L2 regularization method which obtains the size of coefficients by adding the L2 penalty. The ENET [61] is the combination of lasso which uses L1 penalty and ridge which uses L2 penalty. The sizeable number of variables are obtained which helps in avoiding the model turning into excessively sparse model.

The ENET penalty is defined as:

$$\hat{\beta}_{enet} = \underset{\beta}{\operatorname{argmin}} \Bigg[- L(\beta_0, \beta) + \lambda \Bigg(\frac{1-\alpha}{2} \sum_{i=1}^{p} |\beta_i|^2 + \alpha \sum_{i=1}^{p} |\beta_i| \Bigg) \Bigg], \tag{3.15}$$

The smoothly clipped absolute deviation penalty (SCAD) [29] is sparse logistic regression model with a non-concave penalty function. It improves the properties of L1 penalty. The regression coefficients are estimated by minimizing the log-likelihood function:

$$\hat{\beta}_{scad} = \underset{\beta}{\operatorname{argmin}} \Bigg[- L(\beta_0, \beta) + \lambda \sum_{i=1}^{p} p_\lambda(\beta_i) \Bigg], \tag{3.16}$$

In (3.16) the $p_\lambda(\beta_i)$ is defined by:

$$|\beta_i|\mathrm{I}_{(|\beta_i|\leq\lambda)} + \left(\frac{\{(b^2-1)\lambda^2 - (b\lambda - |\beta_i|)_+^2\}I(\lambda \leq |\beta_i|)}{2(b-1)}\right),\ b > 2 \text{ and } \lambda \geq 0\ . \tag{3.17}$$

Minimax concave penalty (MCP) [30] is very similar to the SCAD. However, the MCP relaxes the penalization rate immediately while for SCAD the rate remains smooth before it starts decreasing. The MCP equation is given as follows:

$$\hat{\beta}_{mcp} = \underset{\beta}{\operatorname{argmin}}\left[- L(\beta_0, \beta) + \lambda \sum_{i=1}^{p} p_\lambda(\beta_i)\right]. \tag{3.18}$$

In (3.18) the $p_\lambda(\beta_i)$ is defined as:

$$\Big(\frac{2b\lambda|\beta_i| - \beta_i^2}{2b}\Big)\mathrm{I}(|\beta_i| \leq b\lambda) + \Big(\frac{b\lambda^2}{2}\Big)\mathrm{I}(|\beta_i| > b\lambda),\ \text{for } \lambda \geq 0 \text{ and } b > 1. \tag{3.19}$$

3.2.4 Random Forests

Random Forests (RF) [23] is a simple and interpretive method commonly used for classification purpose in bioinformatics. It is also known for its variable importance ranking in high dimensional data sets. RF is built on the concept of decision trees. Decision trees are usually more decipherable when dealing with binary responses. The idea of RF is to operate as an ensemble instead of relying on single model. RF is a combination of a large number of decision trees where each individual tree has some random subset of features obtained from the data by allowing repetitions. This process is called bagging. The majority voting scheme is applied by aggregating all the tree models and obtain one final prediction.

Given a training set X_i with responses y_i, grow each tree on a independent bootstrap sample from the training data. For $b = 1, ..., B$:, at every node, out of the p variables select m variables at random then detect the best split on these m variables. Output the ensemble of trees $\{f_b\}_1^B$. The predictions from all the individual regression trees on new data x_t can be made by:

$$\hat{f} = \frac{1}{B}\sum_{b=1}^{B} f_b(x_t) \tag{3.20}$$

3.2.5 Support Vector Machines

Support vector machines (SVM) [25] is well known amongst most of the mainstream algorithms in supervised learning. The main goal of SVM is to choose a hyperplane that can best divide the data in the high dimensional space. This helps to avoid overfitting. SVM detects the maximum margin hyperplane, the hyperplane that maximizes the distance between the hyperplane and the closest dots [62]. The maximum margin indicates that the classes are well separable and correctly classified. It is represented as a linear combination of training points. As a result, the decision boundary function for classifying points as to hyperplane only involves dot products between those points. Given the training data, the feature set X_i and class label $y_i \in \{-1, +1\}$. The inner product is defined by:

$$w \cdot x = \sum_{j=1}^{d} w^{(j)} \cdot x^{(j)}. \tag{3.21}$$

For the i-th data point:

$$\gamma_i = (w \cdot x_i + a)y_i \tag{3.22}$$

Solve for γ such that, $\max_{w} \min_{i} \gamma_i$, where the margin of γ_i is as large as possible.
The goal is to find γ such that the margin of the training data is at least γ and written as optimization problem:

$$\begin{aligned} &\max_{w,\gamma} \gamma \\ s.t. &\forall\, i, y_i(w \cdot x_i + a) \geq \gamma \end{aligned} \tag{3.23}$$

3.2.6 Adaboost

Adaboost is also known as adaptive boosting (ABOOST) [24]. It improves the performance of particular weak boosting classifier through an iterative process. This ensemble learning

algorithm can be extensively applied to classification problems. The primary objective here is to assign more weights to the patterns that are harder to classify. Initially, the same weights are assigned to each training item. The weights of the wrongly classified items are incremented while the weights of the rightly classified items are decreased in each iteration. Hence, with the additional iterations and more classifiers, the weak learner is bound to cast on the challenging samples of the training set.

3.3 Evaluation Metrics

We evaluated the results of combinations of FS methods with the classifier using accuracy (Acc) and geometric mean (Gmean). The metrics are detailed with respect to true positive (TP), true negative (TN), false negative (FN), and false positive (FP). The equations for accuracy and Gmean are as follows:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.24}$$

$$\text{Gmean} = \sqrt{Sensitivity \times Specificity}$$

where the sensitivity and specificity are given by:

$$\text{Sensitivity} = \frac{TP}{TP + FN} \quad and \quad \text{Specificity} = \frac{TN}{TN + FP} \tag{3.25}$$

Chapter 4

Results

4.1 Simulation Results

The data is generated based on a random multivariate normal distribution where the mean is assigned as 0 and the variance-covariance matrix $\sum_X$ adapts a compound symmetry structure with the diagonal items set to 1 and the off-diagonal items being ρ values.

$$\sum_X = \begin{pmatrix} 1 & \rho & \cdots & \rho \\ \rho & 1 & \cdots & \rho \\ \vdots & \vdots & \ddots & \vdots \\ \rho & \rho & \cdots & 1 \end{pmatrix}_{p \times p} \tag{4.1}$$

The class labels are generated using the Bernoulli trails with the following probability:

$$\pi_i(y_i = 1|x_i) = \frac{\exp(x_i\beta)}{1 + \exp(x_i\beta)} \tag{4.2}$$

The data matrix $x_i \sim N_p(0, \sum_x)$ is generated using the random multivariate normal distribution and the response variable y_i is generated by binomial distribution as shown in (4.1) and (4.2) respectively. For sufficient comparison of the performance of the model and subsidizing the effects of the data splits, all of the regularized regression models were built using the 10-fold cross validation procedure, the averages were taken over 100 partitioning times referred as 100 iterations in this paper. The data generated are high-dimensional in nature with the number of samples, n = 100 and total features, p = 1000. The true regression coefficients are set to 25 which are generated using uniform distribution with the min and max values 2 and 4, respectively.

With this set up of high-dimensional data, we simulated three different types of data each with correlation structures $\rho = 0.2$, 0.5 and 0.8 respectively. These values show the low, medium and high correlation structures in the data sets which are significantly similar to what we usually see in the gene expression or others among many types of data in the field of bioinformatics.

The prediction performance of any given model is largely dependent on the type of features. The features having an effect on the class will help in attaining the best prediction accuracies. In Figure 4.1 we see the RLFS method with the top-ranked features based on the k-SIS criterion includes the more true number of important features than other existing FS methods such as IG, Chi2, and MRMR used for comparison in this study. The proposed RLFS is performing consistently better across low, medium and highly correlated simulated data and the positive effect of having more number of true important variables is seen in all three simulation scenarios and further explained in detail.

4.1.1 Simulation Scenario (S1): low correlation 0.2

The predictors are generated having a low correlation structure with $\rho = 0.2$. The proposed classifier ERRM is performing better than existing classifier on all the FS methods: proposed RLFS, IG, Chi2 and MRMR. In addition, the proposed combination of RLFS method and ERRM classifier, with the accuracy and Gmean, each of which is 0.8606 and 0.8626 with the standard deviations (SD) of 0.049 and 0.073 respectively, is relatively better in comparison to other combinations of FS method and classifier such as RLFS-LASSO, RLFS-ALASSO, RLFS-ENET and the other remaining combinations as observed in Figure 4.2. The combination of the FS method IG with proposed ERRM with accuracy and SD of 0.8476 and 0.052 is also seen performing better than IG-LASSO, IG-ALASSO, IG-ENET, IG-SCAD, IG-MCP, IG-ABOOST, IG-RF, IG-LR, and IG-SVM. Similarly, the combination of Chi2-ERRM with an accuracy of 0.8538 and a SD of 0.053 is seen better than FS method Chi2 with the other remaining classifiers. The results are reported in Table 4.1. The combination of MRMR-ERRM has an accuracy of 0.8550 and Gmean of 0.8552 is

better than the combination of FS method MRMR with the rest of the nine classifiers. The performance of proposed ERRM shows that ensembles approach is better than using individual classifiers.

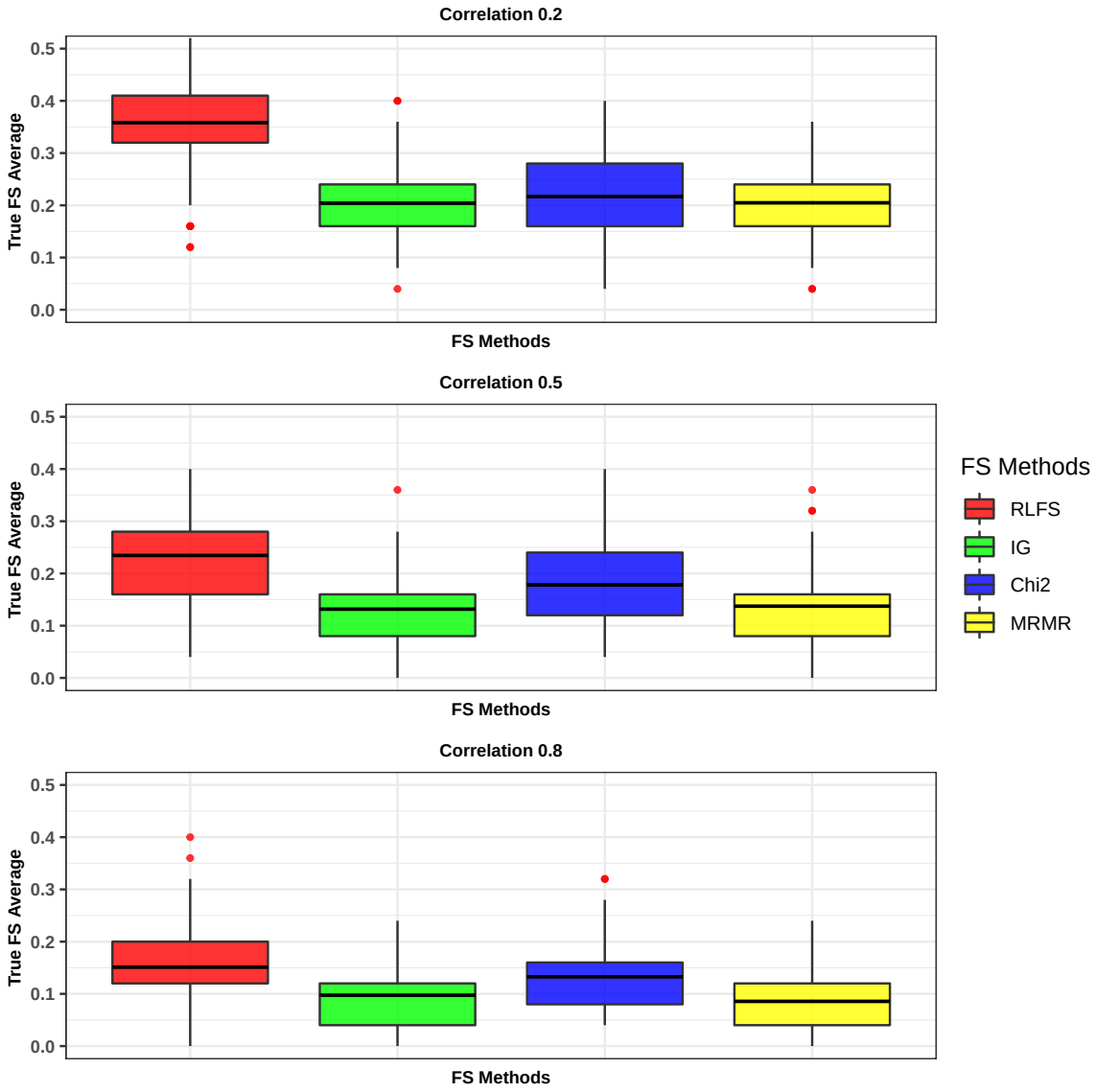

Figure 4.1: *True number of features selected among top k-SIS ranked features and the average of this taken over 100 iterations for three different scenarios*

All the classifiers with the features obtained from RLFS method achieved best accuracies

in comparison to other FS methods. The combination of RLFS with SVM showed the second-best performance. It explains the characteristic of SVM working well with the proposed RLFS as it selected the best features in comparison to other FS methods. In general, the SVM classifier showed very competitive performance with all the FS methods.

The ENET method which is a combination of L1 and L2 penalty showed best performance among all the regularized regression models with all the FS methods, and the best accuracy obtained with RLFS. This can be justified because of the known characteristic of ENET performing well in the high dimensional setting.

In summary, when we compare the average accuracies across all the combinations of FS method and classifiers, the proposed combination of RLFS-ERRM is having better performance than the other existing combinations of the FS and classifier without the proposed FS method RLFS and classifier ERRM itself. For example, the existing FS methods IG, Chi2, and MRMR with the eight existing individual classifiers performance are lower than the proposed RLFS-ERRM combination shown in the Table 4.1.

4.1.2 Simulation Scenario (S2): medium correlation 0.5

The predictor variables are generated using medium correlation structure with $\rho = 0.5$. The proposed combination of RLFS method and ERRM classifier, with the accuracy and Gmean, each of which is 0.9256 and 0.9266 with the standard deviations (SD) of 0.037 and 0.053. respectively, attained relatively better performance compared to other combinations of FS method and classifier such as RLFS-LASSO, RLFS-ALASSO, RLFS-ENET and the other remaining combinations. The results are shown in Table 4.2. From Figure 4.3, we see that the proposed ensemble classifier ERRM with other FS methods such as IG, Chi2, and MRMR is performing best compared to the other nine individual classifiers.

The SVM and ENET classifiers with the RLFS method attained accuracies of 0.9256 and 0.9244 respectively. These accuracies are almost similar to the combination of ERRM-RLFS but the combination of ERRM-RLFS, with the Gmean of 0.9266 and SD of 0.053, outperforms the SVM and ENET classifiers. The average SD of the proposed combination

of the ERRM-RLFS is smaller than other combination of FS method and classifier. This shows the robustness of the proposed combination. The accuracies of SVM and ENET with the IG method are 0.9128 and 0.9150 respectively. These are relatively low compared to the accuracy of 0.9184 achieved by ERRM with IG. Similarly, the ERRM with the Chi2 method having an accuracy of 0.9160 showed relatively better performance than the competitive classifiers ENET and SVM which acquired an accuracy of 0.9122 and 0.9098 respectively. Further, the ERRM classifier with the MRMR method having an accuracy of 0.9174 showed better performance than ENET, SVM, and other top-performing individual classifiers. While the SVM and ENET classifiers showed promising performance on the RLFS that had a good number of important features failed to show the same consistency on the other FS methods where there was more noise. On the other hand, the ensembles ERRM showed robust behavior, with being able to withstand the noise that helps in attaining better prediction accuracies and Gmean, not only with the RLFS method but also with other FS methods such as IG, Chi2, and MRMR.

In summary, when we compare all the combinations of the FS methods and classifiers, the proposed combination of the RLFS-ERRM is seen performing relatively better than any other existing combination of FS and classifiers. The accuracies of the combination of proposed method and other existing methods are very close ranging between 1% to 2%. Similar results are also found in the Simulation Scenario (S3): which is having the highly correlated data with ρ set to 0.8. The results for this scenario are described in the Appendix A.

Table 4.1: Average values taken over 100 iterations in simulation scenario: S1

FS	**Proposed RLFS**		**IG**		**Chi2**		**MRMR**	
+	**Acc**	**Gmean**	**Acc**	**Gmean**	**Acc**	**Gmean**	**Acc**	**Gmean**
Classifier	**(SD)**	**(SD)**	**(SD)**	**(SD)**	**(SD)**	**(SD)**	**(SD)**	**(SD)**
Proposed ERRM	**0.8606 (0.049)**	**0.8626 (0.073)**	0.8476 (0.052)	0.8483 (0.079)	0.8538 (0.053)	0.8551 (0.071)	0.8550 (0.049)	0.8552 (0.075)
LASSO	0.8486 (0.052)	0.8504 (0.075)	0.8316 (0.054)	0.8335 (0.083)	0.8310 (0.052)	0.8323 (0.071)	0.8388 (0.051)	0.8393 (0.077)
ALASSO	0.8402 (0.054)	0.8416 (0.077)	0.8198 (0.051)	0.8217 (0.079)	0.8160 (0.053)	0.8171 (0.075)	0.8304 (0.051)	0.8313 (0.079)
ENET	0.8564 (0.048)	0.8584 (0.072)	0.8424 (0.054)	0.8441 (0.081)	0.8494 (0.046)	0.8509 (0.067)	0.8508 (0.052)	0.8508 (0.077)
SCAD	0.8440 (0.054)	0.8457 (0.080)	0.8264 (0.057)	0.8283 (0.086)	0.8226 (0.061)	0.8239 (0.077)	0.8330 (0.056)	0.8336 (0.081)
MCP	0.8078 (0.049)	0.8095 (0.081)	0.8050 (0.062)	0.8074 (0.088)	0.7936 (0.060)	0.7952 (0.085)	0.8110 (0.060)	0.8126 (0.082)
ABOOST	0.8390 (0.051)	0.8224 (0.077)	0.8314 (0.060)	0.8328 (0.080)	0.8422 (0.054)	0.8435 (0.075)	0.8432 (0.054)	0.8437 (0.075)
RF	0.8432 (0.057)	0.8467 (0.084)	0.8414 (0.052)	0.8435 (0.078)	0.8498 (0.053)	0.8520 (0.075)	0.8522 (0.051)	0.8534 (0.077)
LR	0.8474 (0.050)	0.8489 (0.076)	0.8330 (0.053)	0.8346 (0.080)	0.8370 (0.054)	0.8380 (0.073)	0.8394 (0.051)	0..8394 (0.080)
SVM	0.8582 (0.049)	0.8595 (0.070)	0.8312 (0.052)	0.8320 (0.083)	0.8404 (0.054)	0.8416 (0.074)	0.8388 (0.049)	0.8378 (0.084)

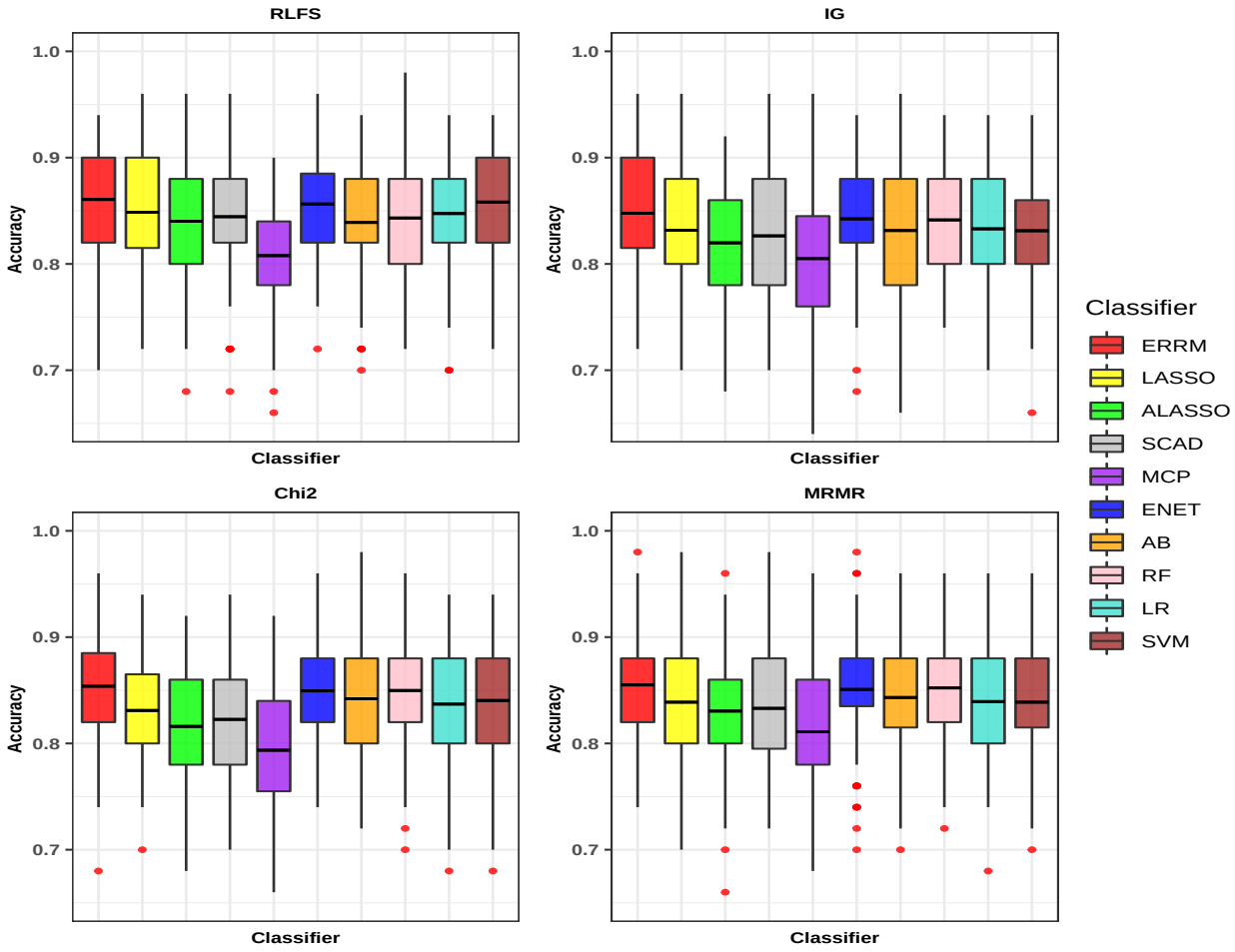

Figure 4.2: *Boxplot showing the accuracies of Classifiers with FS methods in simulation scenario: S1*

Table 4.2: Average values taken over 100 iterations in simulation scenario: S2

FS	**Proposed RLFS**		**IG**		**Chi2**		**MRMR**	
+	**Acc**	**Gmean**	**Acc**	**Gmean**	**Acc**	**Gmean**	**Acc**	**Gmean**
Classifier	**(SD)**	**(SD)**	**(SD)**	**(SD)**	**(SD)**	**(SD)**	**(SD)**	**(SD)**
Proposed ERRM	**0.9256 (0.037)**	**0.9266 (0.053)**	0.9184 (0.039)	0.9195 (0.059)	0.9160 (0.038)	0.9165 (0.056)	0.9174 (0.042)	0.9176 (0.056)
LASSO	0.9146 (0.037)	0.9155 (0.053)	0.9034 (0.045)	0.9046 (0.061)	0.9020 (0.043)	0.9029 (0.063)	0.9066 (0.045)	0.9065 (0.062)
ALASSO	0.9056 (0.039)	0.9062 (0.056)	0.8956 (0.044)	0.8966 (0.065)	0.8948 (0.046)	0.8954 (0.065)	0.8984 (0.046)	0.8982 (0.062)
ENET	0.9244 (0.038)	0.9253 (0.052)	0.9150 (0.044)	0.9163 (0.061)	0.9122 (0.039)	0.9130 (0.060)	0.9158 (0.043)	0.9155 (0.058)
SCAD	0.9102 (0.041)	0.9110 (0.060)	0.8974 (0.046)	0.8986 (0.0630	0.8964 (0.045)	0.8972 (0.065)	0.9030 (0.045)	0.9030 (0.059)
MCP	0.8850 (0.047)	0.8855 (0.066)	0.8798 (0.050)	0.8813 (0.068)	0.8772 (0.045)	0.8782 (0.065)	0.8738 (0.049)	0.8738 (0.070)
ABOOST	0.9158 (0.035)	0.9166 (0.050)	0.9014 (0.046)	0.9027 (0.065)	0.9102 (0.040)	0.9112 (0.060)	0.9072 (0.047)	0.9075 (0.062)
RF	0.9148 (0.039)	0.9166 (0.055)	0.9186 (0.041)	0.9199 (0.059)	0.9154 (0.042)	0.9167 (0.060)	0.9116 (0.043)	0.9127 (0.060)
LR	0.9124 (0.037)	0.9127 (0.054)	0.9054 (0.043)	0.9063 (0.061)	0.9018 (0.045)	0.9024 (0.063)	0.9092 (0.043)	0.9084 (0.060)
SVM	0.9256 (0.038)	0.9261 (0.054)	0.9128 (0.038)	0.9135 (0.056)	0.9098 (0.043)	0.9099 (0.061)	0.9126 (0.045)	0.9120 (0.062)

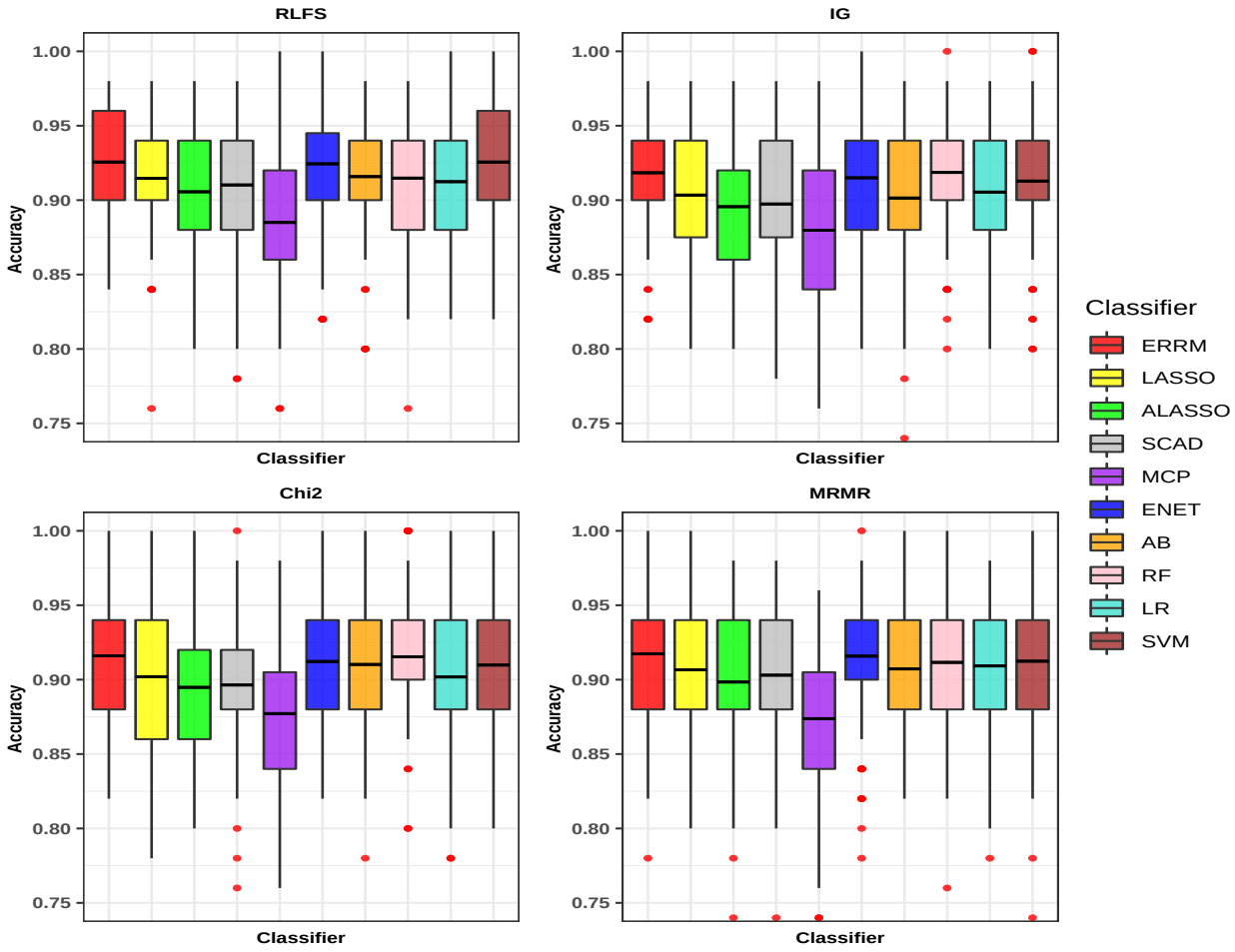

Figure 4.3: *Boxplot showing the accuracies of Classifiers with FS methods in simulation scenario: S2*

4.2 Experimental Results

To test the performance of the proposed combination of ERRM with RLFS, and compare with the rest of the combinations of FS and classifiers, the gene expression data SMK-CAN-187 is analyzed. The data includes 187 samples and 19993 genes obtained from smokers, which included 90 samples with lung cancer and 97 samples without lung cancer. This data is high-dimensional in nature with the number of genes being 19993. The pre-processing

procedures were performed to handle this high-dimensional data. As first filtering step to overcome the redundant noisy features, the marginal maximum likelihood estimation (MMLE) was performed, the genes are ranked based on their level of significance. These ranked significant genes are further applied on the FS methods as the final filtering step and final list of truly significant genes are obtained. These significant genes are applied to all the classification models. The average classification accuracy and Gmean of our proposed framework is tested. At first, the data was divided into training and testing set with 70% and 30% of samples respectively. 70% of the training data is given to the FS methods which rank the genes with respect to their importance and then the top-ranked genes are selected based on k-SIS condition. The selected genes are applied in all the classifiers. For standard comparison and mitigating the effects of the data splitting, all of the regularized regression models were built using the 10-fold cross validation, the models were assessed for testing the performance using different metrics and averages were taken over 100 splitting times referred as 100 iterations.

Figure 4.4 shows the box plot of average accuracies taken over 100 iterations for all the combinations of FS and classifiers. Each of the sub-figures in the figure shows the classifiers with the corresponding FS method. In simulation studies, we set the number of true important significant Chi2 when we generate the synthetic data and then after applying the FS methods, we estimate the average number of true important features as seen in section 3.1. However, the significant genes cannot be known apriori. But, based on the simulation results in section 4.1, where we saw that a higher number of significant features in the model leads to better prediction accuracy. From this, we can see that if the accuracy of classifiers is better then there is a higher number of significant genes in the model. The average number of true important features showed in Figure 4.1 proved that the RLFS method selects more significant features. Likewise, the RLFS has more significant genes than the other FS methods, when we see in Table 4.3, the performance of all the individual classifiers when applied on the RLFS method, the accuracy and Gmean are relatively much better than the accuracies of the individual classifiers when applied on

the IG, Chi2, and MRMR methods.

When we look at the performance of all the classifiers with the IG method in comparison to other FS method, there is much variation in the accuracies. The SVM classifier which attained the accuracy of 0.7026 with the RLFS method drops to 0.6422 with IG method, this shows that the IG is a very poorly performed FS method among all.

The proposed combination of the RLFS with ERRM classifier achieved the highest average accuracy of 0.7161 and Gmean of 0.7127. The RLFS method is also a top-performed FS method on all individual classifiers. However, among the other FS methods, the MRMR method when applied to all the individual classifiers showed relatively much better performance than the application of IG and Chi2 methods to the individual classifiers.

In summary, the proposed combination of the RLFS-ERRM outperformed the rest of the combination of the FS method and classifier. We can also note that the proposed combination is much better than other combinations of the existing FS and classifiers. For example, the proposed RLFS-ERRM is observed to have accuracy of 0.7161 whereas the other combinations such as Chi2-SVM is recorded with 0.6422. This explains that the proposed framework performs much better when the predictor variables are significantly correlated. The second best performed method is the RLFS-ENET combination. The lowest performance among all the combinations of FS method and classifier is shown by SVM classifier with the IG method.

Table 4.3: Average values taken over 100 iterations in experimental data SMK-CAN-187

FS	**Proposed RLFS**		**IG**		**Chi2**		**MRMR**	
+	**Acc**	**Gmean**	**Acc**	**Gmean**	**Acc**	**Gmean**	**Acc**	**Gmean**
Classifier	**(SD)**	**(SD)**	**(SD)**	**(SD)**	**(SD)**	**(SD)**	**(SD)**	**(SD)**
Proposed ERRM	**0.7161 (0.053)**	**0.7127 (0.082)**	0.6789 (0.056)	0.6791 (0.091)	0.6807 (0.056)	0.6808 (0.091)	0.7035 (0.056)	0.7024 (0.087)
LASSO	0.7073 (0.064)	0.7058 (0.087)	0.6726 (0.060)	0.6725 (0.095)	0.6680 (0.057)	0.6680 (0.090)	0.6859 (0.061)	0.6871 (0.097)
ALASSO	0.6878 (0.065)	0.6869 (0.091)	0.6715 (0.060)	0.6714 (0.094)	0.6696 (0.064)	0.6698 (0.092)	0.6800 (0.059)	0.6803 (0.092)
ENET	0.7138 (0.061)	0.7116 (0.085)	0.6733 (0.057)	0.6722 (0.093)	0.6733 (0.052)	0.6726 (0.090)	0.6998 (0.061)	0.6992 (0.095)
SCAD	0.7114 (0.054)	0.7098 (0.083)	0.6735 (0.056)	0.6732 (0.090)	0.6670 (0.058)	0.6669 (0.091)	0.6894 (0.059)	0.6901 (0.091)
MCP	0.6880 (0.010)	0.6870 (0.082)	0.6673 (0.057)	0.6663 (0.089)	0.6647 (0.059)	0.6639 (0.092)	0.6866 (0.057)	0.6874 (0.089)
ABOOST	0.6991 (0.064)	0.6958 (0.087)	0.6673 (0.054)	0.6634 (0.086)	0.6605 (0.058)	0.6583 (0.094)	0.6929 (0.050)	0.6897 (0.083)
RF	0.6975 (0.056)	0.6933 (0.089)	0.6729 (0.045)	0.6691 (0.078)	0.6738 (0.054)	0.6703 (0.090)	0.6942 (0.055)	0.6902 (0.088)
LR	0.7001 (0.065)	0.6987 (0.089)	0.6761 (0.058)	0.6662 (0.097)	0.6770 (0.059)	0.6769 (0.094)	0.7008 (0.058)	0.7000 (0.086)
SVM	0.7026 (0.058)	0.7014 (0.086)	0.6422 (0.059)	0.6430 (0.099)	0.6459 (0.066)	0.6477 (0.105)	0.6668 (0.058)	0.6658 (0.092)

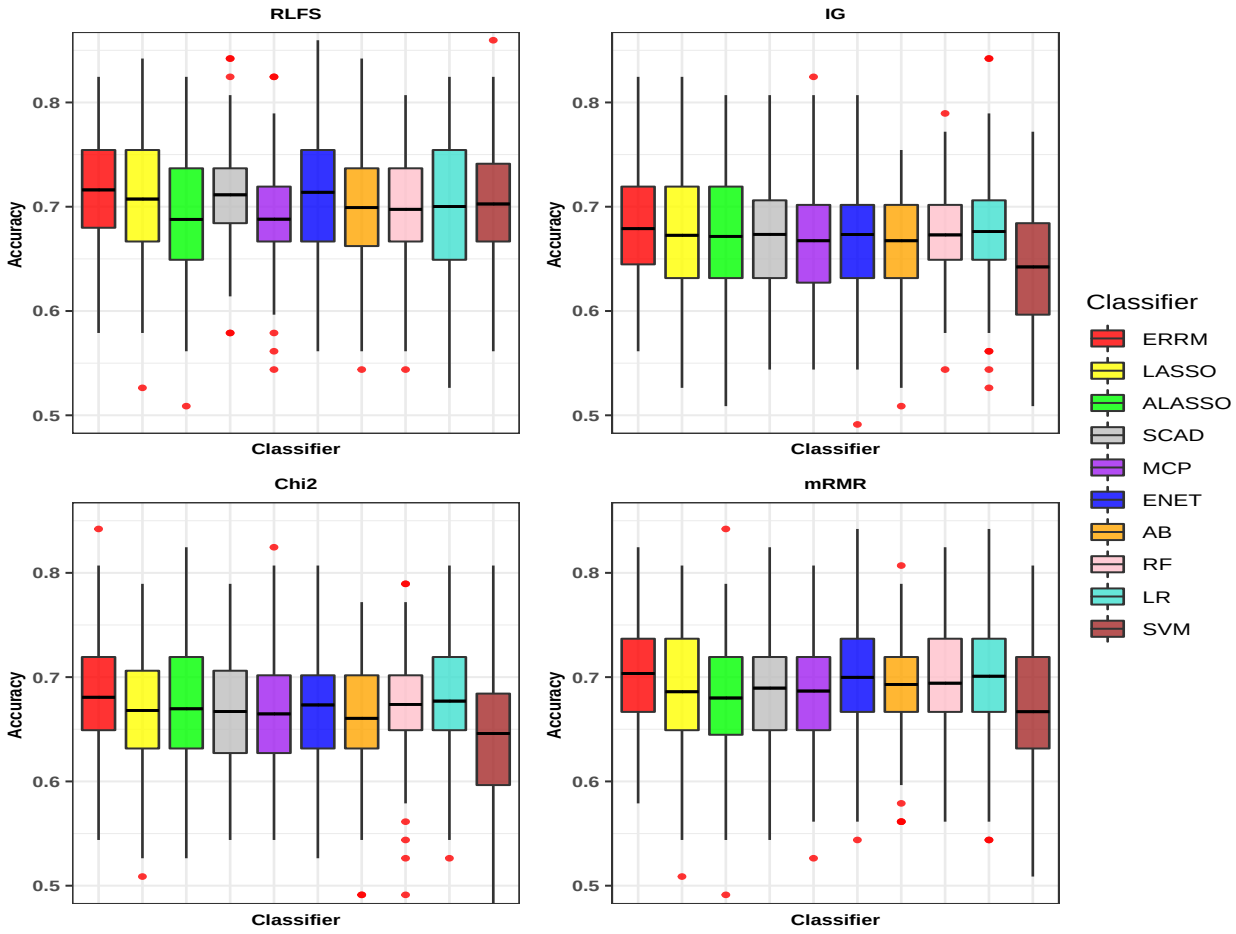

Figure 4.4: *Boxplot showing the accuracies of Classifiers with FS methods in experimental data SMK-CAN-187*

For assessing the importance of bootstrapping and FS screening of the proposed framework, we measured the performance of ERRM without FS screening. The results shows the ensembles method with and without bootstrapping procedure. In the former approach the bootstrapped samples were used and in the latter technique resampled samples were used. In Table 4.4 we see that the ERRM with bootstrapping procedure attains relatively better accuracy in comparison to other ERRM without bootstrapping.

Table 4.4: Comparison of proposed ERRM with and without bootstrapping

	Bootstrapping	Accuracy (SD)	Gmean (SD)
ERRM without FS screening	**Yes**	0.7129 (0.053)	0.7093 (0.091)
ERRM without FS screening	**No**	0.6947 (0.057)	0.6944 (0.089)

The performance of the regularized regression models used in the proposed ensembles algorithm is tested with the FS screening step and without FS screening step. In the former approach, the regularized regression models were built and tested using the proposed RLFS screening step with selected amount of significant features, whereas in the latter approach, the regularized models used all the features for building the model. The FS screening step is necessary for boosting the performance of the model in terms of accuracy and geometric-mean. The results are shown in Table 4.5.

Table 4.5: Comparison of regularized regression models used in the ERRM with and without FS screening

	FS screening	Accuracy (SD)	Gmean (SD)
LASSO	**Yes**	0.7073 (0.064)	0.7058 (0.087)
	No	0.6740 (0.061)	0.6752 (0.125)
ALASSO	**Yes**	0.6878 (0.065)	0.6869 (0.091)
	No	0.6740 (0.061)	0.6752 (0.125)
ENET	**Yes**	0.7138 (0.061)	0.7116 (0.085)
	No	0.6740 (0.061)	0.6752 (0.125)
SCAD	**Yes**	0.7114 (0.054)	0.7098 (0.083)
	No	0.6740 (0.061)	0.6752 (0.125)
MCP	**Yes**	0.6880 (0.010)	0.6870 (0.082)
	No	0.6740 (0.061)	0.6752 (0.125)

Chapter 5

Discussion

In this research, we proposed the combination of RLFS and ERRM that are used for feature selection and classification respectively. The RLFS method ranks the features by employing the lasso method with resampling approach and the k-SIS criteria to set the threshold for selecting the optimal number of features and these features are applied on the ERRM classifier, which uses bootstrapping and rank aggregation, to select the best performing model across the bootstrapped samples and helps in attaining the best prediction accuracy in a high dimensional setting. The ensembles framework ERRM is built using five different regularized regression models. The regularized regression models are known for their best performance in terms of variable selection and prediction accuracy on the gene expression data.

To show the performance of our framework we used three different simulation scenarios with low, medium and high correlation structures that matches the gene expression data. To further illustrate our point, we also used SMK-CAN-187 data. Figure 4.1 shows the boxplots of the average number of true important features, where the RLFS shows higher detection power than the other FS methods such as IG, Chi2, and MRMR. From the results, of both simulation studies and experimental data, we show that all the individual classifiers with RLFS method performed much better compared to the IG, Chi2, and MRMR. We also observed that all the individual classifiers showed a lot of instability with the other three FS methods. This means that the individual classifiers do not work well with more noise and less true important variables in the model. The SVM and ENET classifiers with all the FS methods performed a little better among all the classifiers. However, the performance was relatively still low in comparison to the proposed ERRM classifier with

every FS method. The tree-based ensemble methods RF and ABOOST with RLFS also attained good accuracies but were not the best compared to the ERRM classifier.

The proposed ERRM method is assessed with the FS screening and without FS screening step along with the bootstrapping option. The ERRM with FS screening and bootstrapping approach works better than ERRM without the FS screening and bootstrapping technique. Also, the results from Table 4.4 shows that the ensembles with bootstrapping is better approach on both the filtered and unfiltered data. On comparing the performance of the individual regularized regression models used in the ensembles, the individual models with proposed RLFS screening step showed comparatively better accuracy in comparison to the individual regularized regression models without the FS screening. This means that using the reduced number of significant features with RLFS is better approach instead of using all the features from the data.

The ERRM showed better overall performance not only with the RLFS but also with the other FS methods compared in this study. This means that the ERRM is robust and works much better on the highly correlated gene expression data. The rule of thumb in attaining the best prediction accuracy is that more the true important variables, better the prediction accuracy. Henceforth, from the results of simulation and experimental data we see that the proposed combination of RLFS-ERRM has a superior compared to the other existing combinations of FS and classifiers as seen in the Tables 4.1, 4.2 and 4.3. The proposed ERRM classifier showed the best performance across all the FS methods, with the highest performance achieved with RLFS method. However, the minor drawback in the framework is that the performance of ERRM would have been much better if the RLFS method had better selection average in retaining the significant features.

Chapter 6

Conclusion and Future Work

In this chapter, I conclude my M.S thesis project and propose problems for my Ph.D dissertation.

6.1 Conclusion

In this research, through extensive simulation studies, we showed the superior performance of RLFS with the k-SIS condition has a higher selection average in detecting the true important features than other competitive FS methods. The ensemble classifier ERRM also showed better average prediction accuracy with the RLFS, IG, Chi2, and MRMR compared to other classifiers with these FS methods. On comparing the ensembles, the proposed ERRM with bootstrapping showed better accuracy in comparison to the ERRM without bootstrapping. In both the simulation study and the experimental data SMK-CAN-187, the superior performance was achieved by the proposed combination of RLFS -ERRM compared to all other combinations of FS and classifiers.

6.2 Future Work

As future work, we will further develop methodologies to deal with binary and multi-class classification in high-throughput data.

6.2.1 Data

We will obtain datasets from GEO which are publicly available for testing the performance of our methods. We will also generate the synthetic data to further evaluate the performance of the proposed methods and the existing popular machine learning algorithms.

Simulation data

The synthetic data can be generated in many different ways. We simulate the data with conditions that match the real microarray gene expression data. The microarray data usually have high complex correlation structures, to deal with such data is always challenging. We plan to develop different simulation data types to the real microarray data. Some of the approaches are as follows:

1. Compound Symmetry structure
2. Autoregressive (AR(1)) structure
3. Block correlation structure

In each of these different types of simulation data, we will test the proposed algorithms for different scenarios of the data based on the correlation structures.

Real Data

Because of the advancement of high-throughput technology in biomedical research, large volumes of high-dimensional data are being generated. Some of the classic examples of such data include microarray gene expression data and DNA-methylation data. These data sets are deposited and made publicly available in the Gene expression Omnibus - National Center for Biotechnological Information (GEO - NCBI) website. Several datasets will be obtained from GEO, and these datasets will undergo various pre-processing steps and finally will be used for analyzing the performance of our proposed methods. We also intend to perform a similar procedure for several of the publicly available benchmark datasets.

Some of the gene expression dataset for binary classification includes breast cancer data obtained from GEO with accession number GEO2034 having 286 samples and 22284 genes, colon cancer data with sample size of 62 and 2000 genes, Leukemia data with 72 number of samples and 7129 genes, Lung cancer data from GEO with accession number GSE10072 having 107 samples and 22283 genes, and GSE20189 having 162 samples and 22278 genes. Some of the gene expression data from GEO for multi-class classification problem includes GSE60, GSE1825, and GSE5460.

Dealing with gene expression data is challenging because of the curse of dimensionality. There are several algorithms available to deal with this problem; one of such algorithms include PLR models. One of the other problems of gene expression data includes missing information. The data pre-processing is a crucial step in gene expression data to handle such an issue. Although most of the data are having complete information, because of the nature of experiments conducted, some of the datasets might be having missing information and it becomes a necessity to address this issue. Handling the missing values (MVs) in a high-dimensional setting is an extremely challenging problem. There is a lack of research in this domain; few of the existing approaches for handling MVs are not reliable as there is no complete information about the suitability of those algorithms in handling large datasets. Some of the algorithms for handling MVs are as follows, K-nearest-Neighbor imputation (KNNImpute), Gene Ontology imputation (GOImpute), Multivariate imputations by chained reactions (MICE), regularized and Bayesian lasso regression. We would use one of these approaches for handling the datasets with MVs.

6.2.2 Modified adaptive lasso with proposed weights in high-throughput microarray data

The adaptive lasso method is widely used for variable selection and classification purposes in the high-dimensional setting. In Chapter 4 we described the performance of the adaptive lasso method. The adaptive lasso method uses weights from ridge regression [27]. The

performance of the adaptive lasso can be improved by modifying the weights with different approaches instead of using the standard weights from ridge regression. We plan to propose variants of the adaptive lasso method with new weights from FS and resampling procedures in the high-dimensional setting. In the first variant, the weights will be calculated based on the regression coefficients from the FS methods. We will compare the performance of the FS method with various classifiers including the proposed classification method. In the second variant, we will propose a new weight-based on the resampling technique for the adaptive lasso method. The aim is to compare the performance of the proposed adaptive lasso method with the existing adaptive lasso method. In Chapter 4.1, the compound symmetry structure was introduced for measuring the performance of the proposed framework in this thesis. Develop the simulation study with an auto-regressive structure with different correlation scenarios and apply this data to the proposed framework. We will also find the real high-dimensional microarray data from GEO and measure the performance.

6.2.3 Adaptive K-Nearest-Neighbor with proposed weights in high-dimensional microarray data

In addition to the penalized regression models, random forest, and support vector machines described in Chapter 3.2. KNN method is popularly used for binary classification problems [63]. The KNN method faces the problem of the curse of dimensionality in a high-dimensional setting. We will analyze the performance of the K-Nearest-Neighbor (KNN) algorithm for dealing with such problems by reducing the features using the applicable FS methods [64]. There are also studies conducted on the modification of KNN, adding weights to the method [65]. We plan to propose adaptive KNN with the average radius as weights for dealing with binary classification. The performance of the KNN and the proposed KNN will be tested through simulation and real data applications.

6.2.4 Analyze the performance of FS and classification methods for high-dimensional multi-class classification problem.

We plan to extend the development of the methodology for multi-class classification in high-dimensional data. A two-stage approach is used for dealing with high-dimensional data where the first stage is FS and the second stage is regarded as classification [66, 67]. The popular FS methods in multi-class classification problems include Fisher's score, MRMR, Chi2, Correlation-based FS, Information Gain, Symmetrical Uncertainty, and SVM-RFE. The classification methods such as KNN, Naive Bayes, decision trees, neural networks and support vector machines can be used. The aim is to propose the ensembles of FS and classifiers in a multi-class context. For measuring the performance of the proposed method, firstly, we have to develop a simulation study with a multi-class response. Secondly, we have to find the multi-label microarray data from GEO. With these data sets, The performance of the proposed method will be compared with various existing individual FS and classifiers in multi-class classification problems.

All the algorithms and codes for the developed methodologies will be deposited in an online repository and will be granted access to public for usage.

6.2.5 Timeline

Table 6.1 shows the tentative timeline for the future work to be completed.

Table 6.1: Timeline for completion of this work

Date	Tasks to complete	Goals
Jan. - May. 2020	Modified adaptive lasso with proposed weights in high-throughput microarray data	Submit manuscript
Jun. - Aug. 2020	Adaptive K-Nearest-Neighbor with proposed weights in high-dimensional microarray data	Submit manuscript
Sep. - Dec. 2020	Develop methodology for high-dimensional multi-class classification problem	Present at a conference
Feb. 2021	Complete the first draft of dissertation	Submit first dissertation draft
Mar. 2021	Complete the revised draft of dissertation	Submit revised dissertation draft
Apr. 2021	Defend dissertation	Dissertation defense
May. 2021	Complete the final version of dissertation	Submit final dissertation

References

[1] Hassan Tariq, Elf Eldridge, and Ian Welch. An efficient approach for feature construction of high-dimensional microarray data by random projections. *PLOS ONE*, 13:1–8, 04 2018.

[2] Abhishek Bhola and Shailendra Singh. Gene selection using high dimensional gene expression data: An appraisal. *Current Bioinformatics*, 13(3):225–233, 2018.

[3] Jian J. Dai, Linh H. Lieu, and David M. Rocke. Dimension reduction for classification with gene expression microarray data. *Statistical applications in genetics and molecular biology*, 5:Article6, 2006.

[4] Jun Lu, Robnet T. Kerns, Shyamal D. Peddada, and Pierre R. Bushel. Principal component analysis-based filtering improves detection for affymetrix gene expression arrays. In *Nucleic acids research*, 2011.

[5] Richard Bourgon, Robert Gentleman, and Wolfgang Huber. Reply to talloen et al.: Independent filtering is a generic approach that needs domain specific adaptation. *Proceedings of the National Academy of Sciences*, 107(46):E175–E175, 2010.

[6] Richard Bourgon, Robert Gentleman, and Wolfgang Huber. Independent filtering increases detection power for high-throughput experiments. *Proceedings of the National Academy of Sciences of the United States of America*, 107 21:9546–51, 2010.

[7] Daniel Ramsköld, Eric T. Wang, Christopher B. Burge, and Rickard Sandberg. An abundance of ubiquitously expressed genes revealed by tissue transcriptome sequence data. In *PLoS Computational Biology*, 2009.

[8] Lin Li, Michael Kabesch, Emmanuelle Bouzigon, Florence Demenais, Martin Farrall, Miriam Moffatt, Xihong Lin, and Liming Liang. Using eqtl weights to improve power

for genome-wide association studies: a genetic study of childhood asthma. *Frontiers in Genetics*, 4:103, 2013.

[9] M. L. Calle, V. Urrea, G. Vellalta, N. Malats, and K. V. Steen. Improving strategies for detecting genetic patterns of disease susceptibility in association studies. *Statistics in Medicine*, 27(30):6532–6546, 2008.

[10] Christoph Bock. Analysing and interpreting dna methylation data. *Nature Reviews Genetics*, 13:705–719, 2012.

[11] Hokeun Sun and Shuang Wang. Penalized logistic regression for high-dimensional dna methylation data with case-control studies. *Bioinformatics*, 28 10:1368–75, 2012.

[12] S Kim and S Halabi. High dimensional variable selection with error control. *BioMed Research International*, 2016:1–11, 2016.

[13] Sangjin Kim and Jong-Min Kim. Two-stage classification with sis using a new filter ranking method in high throughput data. *Mathematics*, 7(6), 2019.

[14] Jianqing Fan and Jinchi Lv. Sure independence screening for ultrahigh dimensional feature space. *Journal of the Royal Statistical Society Series B*, 70(5):849–911, November 2008.

[15] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[16] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3:1157–1182, March 2003.

[17] Hanchuan Peng, Fuhui Long, and Chris Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27:1226–1238, 2003.

[18] Gregory Ditzler, J Calvin Morrison, Yemin Lan, and Gail L Rosen. Fizzy: feature subset selection for metagenomics. *BMC bioinformatics*, 16:358, November 2015.

[19] Chao-Ton Su and Chien-Hsin Yang. Feature selection for the svm: An application to hypertension diagnosis. *Expert Systems with Applications*, 34(1):754 – 763, 2008.

[20] Miron B. Kursa and Witold R. Rudnicki. Feature selection with the boruta package. 2010.

[21] Um Okeh and Ica Oyeka. Estimating the fisher's scoring matrix formula from logistic model. 2013.

[22] Ryan J. Urbanowicz, Melissa Meeker, William La Cava, Randal S. Olson, and Jason H. Moore. Relief-based feature selection: Introduction and review. *Journal of biomedical informatics*, 85:189–203, 2017.

[23] Leo Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, October 2001.

[24] Yoav Freund. An adaptive version of the boost by majority algorithm. *Machine Learning*, 43(3):293–318, Jun 2001.

[25] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf. Support vector machines. *IEEE Intelligent Systems and their Applications*, 13(4):18–28, July 1998.

[26] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996.

[27] Donald W. Marquardt and Ronald D. Snee. Ridge regression in practice. *The American Statistician*, 29(1):3–20, 1975.

[28] Xin-Guang Yang and Yongjin Lu. Informative gene selection for microarray classification via adaptive elastic net with conditional mutual information. *ArXiv*, abs/1806.01466, 2018.

[29] Jianqing Fan and Runze Li. Variable selection via nonconcave penalized likelihood and its oracle properties. 2001.

[30] Cun-Hui Zhang. Nearly unbiased variable selection under minimax concave penalty. *The Annals of Statistics*, 38(2):894–942, Apr 2010.

[31] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference and prediction*. Springer, 2 edition, 2009.

[32] Thomas G. Dietterich. Ensemble methods in machine learning. In *Proceedings of the First International Workshop on Multiple Classifier Systems*, MCS '00, pages 1–15, London, UK, UK, 2000. Springer-Verlag.

[33] Richard Maclin and David W. Opitz. Popular ensemble methods: An empirical study. *CoRR*, abs/1106.0257, 2011.

[34] Leo Breiman. Bagging predictors. *Mach. Learn.*, 24(2):123–140, August 1996.

[35] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, 55(1):119–139, August 1997.

[36] Susmita Datta, Vasyl Pihur, and Somnath Datta. An adaptive optimal ensemble classifier via bagging and rank aggregation with applications to high dimensional data. In *BMC Bioinformatics*, 2010.

[37] Hongshik Ahn, Hojin Moon, Melissa J. Fazzari, Noha Lim, James J. Chen, and Ralph L. Kodell. Classification by ensembles from random partitions of high-dimensional data. *Computational Statistics Data Analysis*, 51:6166–6179, 2007.

[38] Noha Lim, Hongshik Ahn, Hojin Moon, and James J. Chen. Classification of high-dimensional data with ensemble of logistic regression models. *Journal of biopharmaceutical statistics*, 20 1:160–71, 2009.

[39] Igor Kononenko. Estimating attributes: Analysis and extensions of relief. In *ECML*, 1994.

[40] Roberto Battiti. Using mutual information for selecting features in supervised neural net learning. *IEEE transactions on neural networks*, 5 4:537–50, 1994.

[41] François Fleuret. Fast binary feature selection with conditional mutual information. *J. Mach. Learn. Res.*, 5:1531–1555, 2004.

[42] Mehdi Pirooznia, Jack Y. Yang, Mary Qu Yang, and Youping Deng. A comparative study of different machine learning methods on microarray gene expression data. *BMC Genomics*, 9:S13 – S13, 2008.

[43] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2008. ISBN 3-900051-07-0.

[44] Andy Liaw and Matthew Wiener. Classification and regression by randomforest. *R News*, 2(3):18–22, 2002.

[45] Miron B. Kursa. *praznik: Collection of Information-Based Feature Selection Filters*, 2018. R package version 5.0.0.

[46] Frank Pessler Frank Klawonn Natalia Novoselova, Junxi Wang. *Biocomb: Feature Selection and Classification with the Embedded Validation Procedures for Biomedical Data Analysis*, 2018. R package version 0.4.

[47] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010.

[48] Patrick Breheny and Jian Huang. Coordinate descent algorithms for nonconvex penalized regression, with applications to biological feature selection. *Annals of Applied Statistics*, 5(1):232–253, 2011.

[49] David Meyer, Evgenia Dimitriadou, Kurt Hornik, Andreas Weingessel, and Friedrich Leisch. *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien*, 2019. R package version 1.7-1.

[50] Vasyl Pihur, Somnath Datta, and Susmita Datta. *RankAggreg: Weighted Rank Aggregation*, 2018. R package version 0.6.5.

[51] Feature Selection Datasets. http://featureselection.asu.edu/old/datasets.php. Accessed: 13-Oct-2019.

[52] Verónica Bolón-Canedo, Noelia Sánchez-Maroño, Amparo Alonso-Betanzos, José Manuel Benítez, and Francisco Herrera. A review of microarray datasets and applied feature selection methods. *Inf. Sci.*, 282:111–135, 2014.

[53] Mingyuan Wang and Adrian Barbu. Are screening methods useful in feature selection? an empirical study. In *PloS one*, 2018.

[54] Jundong Li, Kewei Cheng, Suhang Wang, Fred Morstatter, Robert P. Trevino, Jiliang Tang, and Huan Liu. Feature selection: A data perspective. *ACM Comput. Surv.*, 50:94:1–94:45, 2017.

[55] In-Hee Lee, Gerald H. Lushington, and Mahesh Visvanathan. A filter-based feature selection approach for identifying potential biomarkers for lung cancer. In *Journal of Clinical Bioinformatics*, 2010.

[56] Borja Seijo-Pardo, Verónica Bolón-Canedo, and Amparo Alonso-Betanzos. Using a feature selection ensemble on dna microarray datasets. In *ESANN*, 2016.

[57] Xin Jin, Anbang Xu, Rongfang Bie, and Ping Guo. Machine learning techniques and chi-square feature selection for cancer classification using sage gene expression profiles. In *BioDM*, 2006.

[58] Xiao Dong Chen and Hemant Ishwaran. Random forests for genomic data analysis. *Genomics*, 99 6:323–9, 2012.

[59] Concha Bielza, V. Robles, and Pedro Larrañaga. Regularized logistic regression without a penalty term: An application to cancer classification with microarray data. *Expert Syst. Appl.*, 38:5110–5118, 2011.

[60] J. G. Liao and Khew-Voon Chin. Logistic regression for disease classification using microarray data: model selection in a large p and small n case. *Bioinformatics*, 23 15:1945–51, 2007.

[61] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 67(2):301–320, 2005.

[62] Li Ying, Zhang Yu, and Zhao Shishun. Gender classification with support vector machines based on non-tensor pre-wavelets. *2010 Second International Conference on Computer Research and Development*, pages 770–774, 2010.

[63] R. Mitch Parry, W. Jones, Todd H. Stokes, Jeff Phan, Richard A. Moffitt, Hengfu Fang, L Shi, Aloys Oberthuer, Margit Bistrup Fischer, Weida Tong, and May D. Wang. k-nearest neighbor models for microarray gene expression analysis and clinical outcome prediction. In *The Pharmacogenomics Journal*, 2010.

[64] Mukesh Kumar, Nitish Kumar Rath, Amitav Swain, and Santanu Kumar Rath. Sciencedirect eleventh international multi-conference on information processing-2015 (imcip-2015) feature selection and classification of microarray data using mapreduce based anova and k-nearest neighbor. 2015.

[65] Klaus Hechenbichler and Klaus Schliep. Weighted k-nearest-neighbor techniques and ordinal classification. 2004.

[66] Junshan Yang, Jiarui Zhou, Zexuan Zhu, Xiaoliang Ma, and Zhen Ji. Iterative ensemble feature selection for multiclass classification of imbalanced microarray data. In *Journal of Biological Research-Thessaloniki*, 2016.

[67] Tao Li, Chengliang Zhang, and Mitsunori Ogihara. A comparative study of feature selection and multiclass classification methods for tissue classification based on gene expression. *Bioinformatics*, 20 15:2429–37, 2004.

Appendix A

Simulation Scenario: S3

The data is generated based on high correlation data structure with $rho = 0.8$. The performance of proposed combination of RLFS-ERRM is relatively better than the other combinations of the FS methods and classifiers. The results for simulation scenario S3 are shown in Figure A.1. The average accuracy and Gmean for all the FS and classifiers are noted in the Table A.1. The SVM and ENET classifiers with all the FS methods showed a little better performance among all individual the classifiers. However, the accuracy and Gmean attained by the proposed ensemble classifier ERRM with the FS methods RLFS, IG and Chi2 was relatively better compared to the individual classifiers with FS methods. In summary, the best performance is achieved by the proposed RLFS-ERRM combination with accuracy of 0.9586 and Gmean of 0.9596. The second best performing combination is MRMR-SVM. The lowest performance in terms of accuracy and the Gmean is shown by Chi2-MCP. The MCP classifier is having the lowest accuracies with all the FS methods. This explains that the MCP does not perform well when the predictor variables are highly correlated

Table A.1: Average values taken over 100 iterations in simulation scenario: S3

FS	**Proposed RLFS**		**IG**		**Chi2**		**MRMR**	
+	**Acc**	**Gmean**	**Acc**	**Gmean**	**Acc**	**Gmean**	**Acc**	**Gmean**
Classifier	**(SD)**	**(SD)**	**(SD)**	**(SD)**	**(SD)**	**(SD)**	**(SD)**	**(SD)**
Proposed ERRM	**0.9586 (0.025)**	**0.9596 (0.039)**	0.9556 (0.027)	0.9565 (0.041)	0.9530 (0.034)	0.9544 (0.045)	0.9560 (0.024)	0.9558 (0.037)
LASSO	0.9482 (0.033)	0.9493 (0.050)	0.9442 (0.030)	0.9194 (0.045)	0.9428 (0.037)	0.9447 (0.051)	0.9444 (0.032)	0.9442 (0.042)
ALASSO	0.9420 (0.031)	0.9425 (0.051)	0.9376 (0.030)	0.9379 (0.047)	0.9328 (0.041)	0.9342 (0.056)	0.9388 (0.033)	0.9389 (0.047)
ENET	0.9576 (0.025)	0.9587 (0.039)	0.9538 (0.029)	0.9546 (0.042)	0.9532 (0.034)	0.9546 (0.045)	0.9566 (0.024)	0.9562 (0.036)
SCAD	0.9464 (0.031)	0.9475 (0.049)	0.9422 (0.030)	0.9428 (0.045)	0.9386 (0.043)	0.9401 (0.055)	0.9414 (0.031)	0.9408 (0.043)
MCP	0.9256 (0.040)	0.9270 (0.062)	0.9262 (0.038)	0.9269 (0.055)	0.9210 (0.041)	0.9221 (0.058)	0.9224 (0.034)	0.9223 (0.048)
ABOOST	0.9454 (0.032)	0.9469 (0.047)	0.9494 (0.030)	0.9501 (0.044)	0.9470 (0.034)	0.9482 (0.046_	0.9480 (0.029)	0.9481 (0.040)
RF	0.9540 (0.030)	0.9557 (0.043)	0.9560 (0.029)	0.9565 (0.043)	0.9542 (0.032)	0.9556 (0.044)	0.9508 (0.027)	0.9510 (0.039)
LR	0.9478 (0.029)	0.9482 (0.045)	0.9462 (0.030)	0.9469 (0.044)	0.9418 (0.038)	0.9432 (0.050)	0.9438 (0.028)	0.9437 (0.041)
SVM	0.9560 (0.027)	0.9568 (0.041)	0.9522 (0.030)	0.9527 (0.043)	0.9520 (0.031)	0.9526 (0.042)	0.9594 (0.026)	0.9587 (0.037)

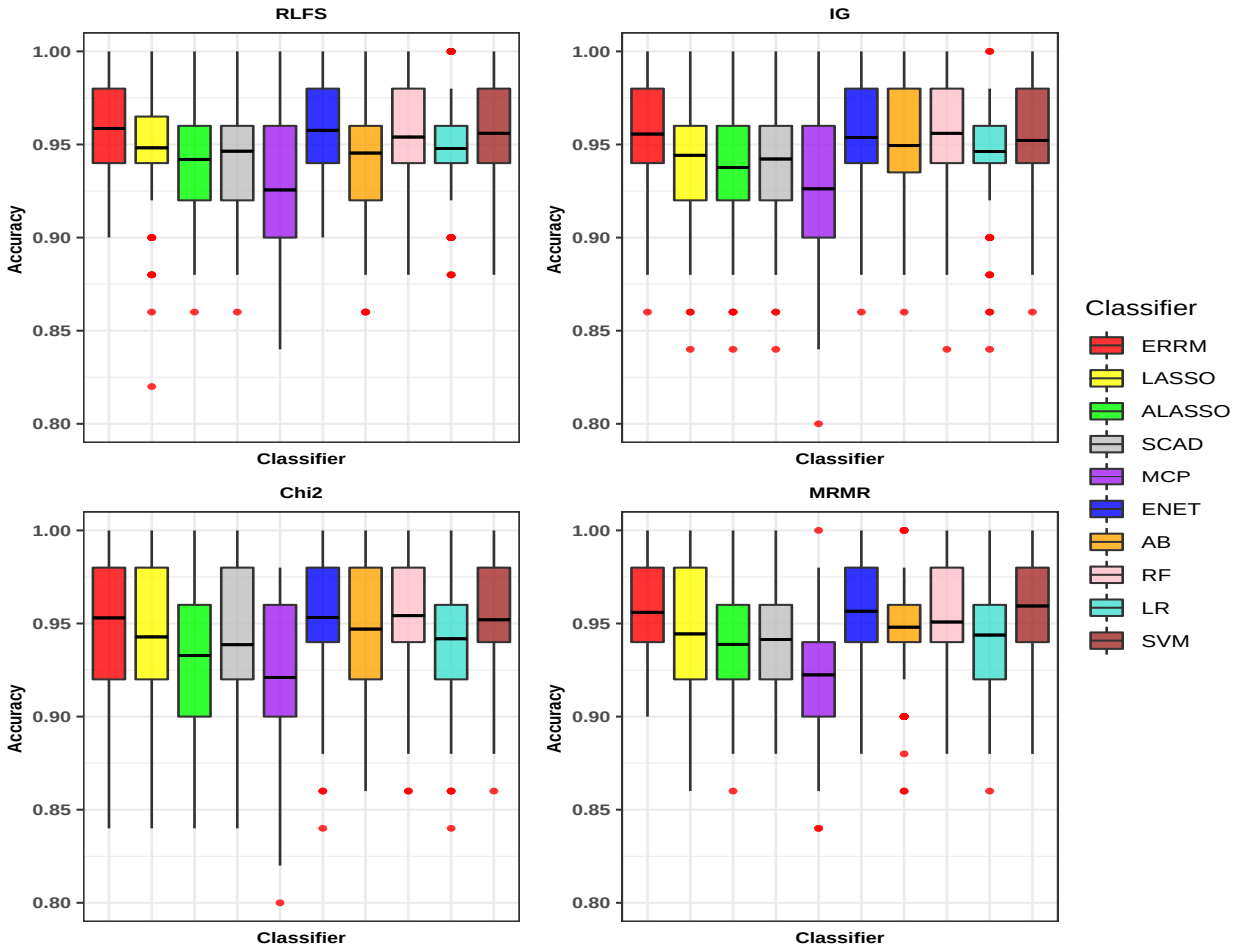

Figure A.1: *Boxplot showing the accuracies of Classifiers with FS methods in simulation scenario: S3*

Appendix B

The RLFS-ERRM Program

```
library(mvtnorm)
library(doParallel)
library(class)
library(adabag)
library(tidyverse)
library(Biocomb)
library(praznik)
library(randomForest)
library(glmnet)
library(ncvreg)
library(e1071)
library(caret)
library(RankAggreg)
library(ggplot2)

## Functions to calculate performance metrics
## accuracy
accuracy <- function(truth, predicted)
{
  if(length(truth) > 0)
    sum(truth==predicted)/length(truth) else
```

```
        return(0)
}
## sensitivity
sensitivity <- function(truth, predicted)
{
  if(sum(truth==1) > 0)
    sum(predicted[truth==1]==1)/sum(truth==1)    else
        return(0)
}
## specificity
specificity <- function(truth, predicted)
{
  if(sum(truth==0) > 0)
    sum(predicted[truth==0]==0)/sum(truth==0)    else
        return(0)
}

## generate the feature set
feature_set = function(nsample, p, seed.index, beta, cors )
  {

    set.seed(seed.index)
    cor.mat = matrix(1, nrow = p, ncol = p)
    cor.mat[lower.tri(cor.mat)]=cors
    # cor.mat[upper.tri(cor.mat)]=cors
    cor.mat=t(cor.mat)
    cor.mat[lower.tri(cor.mat)] = cors
    mu.vec=rep(0,p)
```

```
    dat = rmvnorm(n = nsample, mean = mu.vec,
    sigma = cor.mat, method = "svd")
    # change to eigen or chol or svd if needed
    return(dat)
  }

## generate response
generate_response_data = function(nsample, p, cors, theta0)
  {
    cnt      = 0
    case.dat = NULL
    control.dat = NULL
    # generate response of equal control and case
    while(cnt==0)
    {
      seed.val = sample(1:1000000,1)
      data.set = feature_set(nsample, p, seed.val, beta, cors)
      data.set = scale(data.set)
      feta = data.set%*%theta0
      fprob = exp(feta)/(1+exp(feta))
      y = rbinom(nsample, 1, fprob)
      idx = which(y == 1)

      if((length(idx)==(nsample/2)))
      #& (length(idx)<=((nsample/2)+20)))
      {
        result = list(y,data.set,seed.val)
        cnt    = 100
```

```
        return(result)
      }
    }
  }

data.gen <- function()
{
  nsample = 200;          cors     = 0.2;
  p = 1000;               num.beta = 25
  beta  = runif(num.beta,2,4)
  true.coefs            = beta
  true.coefs.idx        = c(1:num.beta)
  # indices of the true regression coefs
  theta0                = rep(0,p)
  # generate the empty list of values
  theta0[c(true.coefs.idx)] = c(true.coefs)

  obj <- generate_response_data(nsample = nsample, p,
  cors =  cors, theta0 = theta0)

  y1       = obj[[1]]  # response variable
  x1       = data.frame(obj[[2]]) # data set
  df <- data.frame(x1,y1)
  colnames(df)=paste("v",1:ncol(df),sep="")
  colnames(df)[ncol(df)]<- "y"
  true.var.names = paste("v",c(true.coefs.idx),sep="")

  return(df)
```

```
}

## real data generation function
data.gen <- function()
{
  data.SMK_CAN_187 <- readMat("SMK_CAN_187.mat")
  x1 <- as.matrix(data.SMK_CAN_187$X)
  y1 <- as.numeric(data.SMK_CAN_187$Y)
  y1 <- replace(y1, y1==1, 0)
  y1 <- replace(y1, y1==2, 1)

  s.idx <- sample(1:nrow(x1),as.integer(nrow(x1)*.7),replace = F)

  train.y = y1[s.idx]
  train.x = x1[s.idx,]
  test.x =  x1[-s.idx,]
  test.y =  y1[-s.idx]

  data.train <- data.frame(train.x, train.y)
  data.test <- data.frame(test.x, test.y)

  return(list(data.train, data.test))
}

## Function to convert scores
convertScores <- function(scores)
{
  scores <- t(scores)
```

```
  ranks <- matrix(0, nrow(scores), ncol(scores))
  weights <- ranks
  for(i in 1:nrow(scores)){
    ms <- sort(scores[i,], decr=TRUE, ind=TRUE)
    ranks[i,] <- colnames(scores)[ms$ix]
    weights[i,] <- ms$x
  }
  list(ranks = ranks, weights = weights)
}
## Function to calculate the PLR models
## lasso
lasso = function(x,y)
{
  cv.lasso <- cv.glmnet(y = y, x = x, alpha=1, family="binomial")
  return(cv.lasso)
}
## alasso
alasso = function(x,y)
{
  ## Ridge Regression to create the Adaptive Weights Vector
  cv.ridge <- cv.glmnet(y = y, x = x, alpha=0, family='binomial')
  w3 <- 1/abs(matrix(coef(cv.ridge,
  s=cv.ridge$lambda.min)[, 1][2:(ncol(x)+1)] ))^1
  ## Using gamma = 1
  w3[w3[,1] == Inf] <- 999999999
  ## Replacing values estimated as Infinite for 999999999

  ## Adaptive Lasso
```

```
  cv.alasso <- cv.glmnet(y = y, x = x, alpha=1,
  family="binomial", penalty.factor = w3)
  return(cv.alasso)
}
## bridge
enet = function(x,y)
{
  cv.enet <- cv.glmnet(y=y, x= x, alpha=0.5, family="binomial")
  return(cv.enet)
}
## scad
scad = function(x,y)
{
  cv.scad <- cv.ncvreg(x, y,family="binomial",penalty="SCAD")
  return(cv.scad)
}

## mcp
mcp = function(x,y)
{
  cv.mcp <- cv.ncvreg(x, y,family="binomial",penalty="MCP")
  return(cv.mcp)
}
## Function to perform feature selection
feature.selection = function(method, data)
{
  df <- data
```

```
## Split the data into training and testing set
bound <- floor((nrow(df)/4)*3)
df <- df[sample(nrow(df)), ]    #sample rows
df.train <- df[1:bound, ]       #get training set
df.test <- df[(bound+1):nrow(df), ]

##-----------------------------------------------------------
train.x <-data.matrix(df.train[,-(ncol(df.train))])
train.y <- as.numeric(df.train[,(ncol(df.train))])
# Test set seperated into data matrix and response variable
test.x <- data.matrix(df.test[,-(ncol(df.test))])
test.y <- as.numeric(df.test[,(ncol(df.test))])

data.train <- data.frame(train.x, train.y)
# Last column name is train.y
data.test <- data.frame(test.x, test.y)
# Last column name is test.y

if(method == "resampling")
{
  final_lasso <- main(train.x = train.x, train.y = train.y,
  method = "lasso")
  set_lasso <- names(final_lasso[1:top])
  idx <- match(set_lasso, names(data.train))
}
if(method=="information")
{
  data.train[,ncol(data.train)]
```

```
  <- as.factor(data.train[,ncol(data.train)])
   disc<-"equal interval width"
   attrs.nominal <- numeric()
   fit.info =
   select.inf.gain(data.train,disc.method=
   disc,attrs.nominal=attrs.nominal)
   idx <- as.integer(fit.info$NumberFeature)[1:top]
}
if(method=="chi2")
{
   data.train[,ncol(data.train)]<-as.factor
   (data.train[,ncol(data.train)])
   disc<-"equal interval width"
   attrs.nominal <- numeric()
   fit.chi2 <- select.inf.chi2(data.train,disc.method=
   disc,attrs.nominal=attrs.nominal)
   idx <- as.integer(fit.chi2$NumberFeature)[1:top]

}
if(method=="mrmr")
{
   fit.mrmr = MRMR(data.train[,-ncol(data.train)],
   data.train[,ncol(data.train)],k=ncol(data.train
   [,-ncol(data.train)]))
   fm<-fit.mrmr$score
   set <- names(fm)[1:top]
   idx <- match(set, names(data.train))
```

```
  }
  return(list(data.train, data.test, idx))
}

## Function to calculate Resampling method
freq.matrix = function(train.x,train.y,method)
{
  if(method=="lasso")
  {
    fit = glmnet(as.matrix(train.x),train.y,alpha=1,
    family="binomial")
    beta.matrix = fit$beta
  }
  if(method=="bridge")
  {
    fit = glmnet(as.matrix(train.x),train.y,alpha=(1/2),
    family="binomial")
    beta.matrix = fit$beta
  }

  if(method=="scad")
  {
    fit = ncvreg(as.matrix(train.x),
    train.y,family="binomial",penalty="SCAD")
    beta.matrix = fit$beta[-c(1),]
  }

  if(method=="mcp")
```

```
  {
    fit = ncvreg(as.matrix(train.x),
     train.y,family="binomial",penalty="MCP")
    beta.matrix = fit$beta[-c(1),]
  }
  result.matrix = coefs.matrix = matrix(0,nrow=ncol
  (beta.matrix),
  ncol=nrow(beta.matrix))
  colnames(result.matrix) = rownames(beta.matrix)
  for(i in 1:nrow(result.matrix))
  {
    idx = which(beta.matrix[,i]!=0)
    result.matrix[i,idx] = 1
    coefs.matrix[i,idx] = beta.matrix[idx,i]
  }
  # return frequency matrix
  return(list(result.matrix,coefs.matrix))
}

resampling = function(train.x,train.y,method)
{
  idx = sample(1:nrow(train.x),as.integer(nrow(train.x)*0.7))
  s.x = train.x[idx,]
  s.y = train.y[idx]

  if(method=="lasso")
  {
    obj = freq.matrix(s.x,s.y,method)
```

```
  }
  if(method=="bridge")
  {
    obj = freq.matrix(s.x,s.y,method)
  }
  if(method=="scad")
  {
    obj = freq.matrix(s.x,s.y,method)
  }
  if(method=="mcp")
  {
    obj = freq.matrix(s.x,s.y,method)
  }
  # return frequency matrix and coeficients matrix
  return(obj)
}

add = function(obj.mat,option)
{
  if(option=="frequency"){s = apply(obj.mat,2,sum)}
  if(option=="coef"){s = apply(obj.mat,2,function(x)max(abs(x)))}
  return(s)
}
##########################################################################
resampling_results = function(train.x,train.y,method)
{
  # 100 resampling
  # return frequency and coefficients
```

```
  # for each variables
  mat0 = matrix(0,nrow=100,ncol=ncol(train.x))
  mat1 = matrix(0,nrow=100,ncol=ncol(train.x))
  colnames(mat0) = colnames(mat1) = colnames(train.x)
  lens = NULL
  for(i in 1:100)  ##############################################
  {
    # lasso
    obj0 = resampling(train.x,train.y,method=method)
    lens = c(lens,nrow(obj0[[1]]))
    mat0[i,] = add(obj0[[1]],"frequency")
    mat1[i,] = add(obj0[[2]],"coef")
  }
  freq.vec = apply(mat0,2,sum)/sum(lens)
  coef.vec = apply(mat1,2,function(x)max(abs(x)))
  return(list(freq.vec,coef.vec))
}

main = function(train.x,train.y,method)#,directory)
{
  results = resampling_results(train.x,train.y,method)
  mu.vec      = results[[1]]
  max.coefs.vec     = results[[2]]

  s.mu.vec = sort(mu.vec,decreasing=T,index.return=T)
  rank.mu.vec     = mu.vec[s.mu.vec$ix]
  return(rank.mu.vec)
}
```

```
test.function <- function(feature.idx, method)
{
  fs <- feature.idx
  data.train <- fs[[1]]
  data.test <- fs[[2]]
  idx <- fs[[3]]
  ######################################
  reduced.trainx <- data.train[,idx]
  reduced.trainy <- as.factor(data.train[,ncol(data.train)])

  test.x <- data.test[,idx]
  test.y <- as.factor(data.test[,(ncol(data.test))])
  ######################################
  dat.train <- data.frame(reduced.trainx, reduced.trainy)
  ######################################
  n <- length(reduced.trainy)
  nalg <- length(algorithms)
  nvm <- length(validation)
  ly <- levels(reduced.trainy)

  fittedModels <- list() #to keep the fitted algorithms
  coefs.mat = freq.mat = matrix(0,nrow=M,
  ncol=ncol(reduced.trainx))
  colnames(coefs.mat)=colnames(freq.mat)=
  colnames(reduced.trainx)
  best.methods=NULL
  mat.result = matrix(0,nrow=nrow(test.x),ncol=M)
```

```
if(method=="ensemble")
{
  for(k in 1:M)
  {
    repeat
    { # to make sure that at least more than
      # two classes are present
      s <- sample(n, replace=TRUE)
      if(length(table(reduced.trainy[s])) >= 2
     & length(table(reduced.trainy[-s])) >= 2)
        break
    }

    fs <- 1:ncol(reduced.trainx)
    sub.trainingx <- reduced.trainx[s, fs]
    sub.testingx  <- reduced.trainx[-unique(s), fs]
    sub.trainingy <- reduced.trainy[s]
    sub.testingy <- reduced.trainy[-unique(s)]

    predicted <- list()
    Res <- list()
    # train all algorithms on the subset
    # Fit the model and predict
    for(j in 1:nalg)
    {
      Res[[j]]<- switch(algorithms[j],
```

```
                    "lasso"  =  {

                        fit.lasso = lasso(as.matrix
                        (sub.trainingx), sub.trainingy)
                        lasso.idx = which(abs(coef(fit.lasso)
                        [,1])>0)[-c(1)]
                        lasso.m.idx = match(names(lasso.idx),
                        colnames(sub.testingx))
                        lasso.xb =
                        as.matrix(cbind(1,as.matrix
 (sub.testingx[,lasso.m.idx])))%*%as.matrix
          (coef(fit.lasso)[c(1,lasso.idx),1])
     lasso.pred = exp(lasso.xb)/(1+exp(lasso.xb))
     lasso.pred <- ifelse(lasso.pred>0.5, 1, 0)
     predicted[[j]] <- as.character(lasso.pred[,1])

                    },
                    "alasso"  =  {

                        fit.alasso = alasso(as.matrix
                        (sub.trainingx), sub.trainingy)
alasso.idxwhich(abs(coef(fit.alasso)[,1])>0)[-c(1)]
alasso.m.idx =
match(names(alasso.idx),colnames(sub.testingx))
          alasso.xb = as.matrix(cbind(1,
          as.matrix(sub.testingx[,alasso.m.idx])))
         %*%as.matrix(coef(fit.alasso)[c(1,alasso.idx),1])
alasso.pred = exp(alasso.xb)/(1+exp(alasso.xb))
```

```
alasso.pred <- ifelse(alasso.pred>0.5, 1, 0)
predicted[[j]] <- as.character(alasso.pred[,1])

                    },
                    "bridge"  = {

fit.enet = enet(as.matrix(sub.trainingx), sub.trainingy)
enet.idx = which(abs(coef(fit.enet)[,1])>0)[-c(1)]
enet.m.idx = match(names(enet.idx),
colnames(sub.testingx))
enet.xb = as.matrix(cbind(1,as.matrix(
sub.testingx[,enet.m.idx])))%*%as.matrix
(coef(fit.enet)[c(1,enet.idx),1])
enet.pred = exp(enet.xb)/(1+exp(enet.xb))
enet.pred <- ifelse(enet.pred>0.5, 1, 0)
predicted[[j]] <- as.character(enet.pred[,1])

                    },
                    "scad"  = {

fit.scad  = scad(as.matrix(sub.trainingx), sub.trainingy)
scad.idx = which(abs(coef(fit.scad))>0)[-c(1)]
scad.m.idx = match(names(scad.idx),
colnames(sub.testingx))
scad.xb = as.matrix(cbind(1,as.matrix(
sub.testingx[,scad.m.idx])))%*%as.matrix(
coef(fit.scad)[c(1,scad.idx)])
scad.pred = exp(scad.xb)/(1+exp(scad.xb))
```

```
        scad.pred <- ifelse(scad.pred>0.5, 1, 0)
        predicted[[j]] <- as.character(scad.pred[,1])

                                },
                                "mcp"   = {

        fit.mcp   = mcp(as.matrix(sub.trainingx), sub.trainingy)
        mcp.idx = which(abs(coef(fit.mcp))>0)[-c(1)]
        mcp.m.idx = match(names(mcp.idx),colnames(sub.testingx))
mcp.xb = as.matrix(cbind(1,as.matrix(
sub.testingx[,mcp.m.idx])))%*%as.matrix
(coef(fit.mcp)[c(1,mcp.idx)])
        mcp.pred = exp(mcp.xb)/(1+exp(mcp.xb))
        mcp.pred <- ifelse(mcp.pred>0.5, 1, 0)
        predicted[[j]] <- as.character(mcp.pred[,1])

                                }
        )
        attr(Res[[j]], "algorithm") <- algorithms[j]
      }

      # compute validation measures
      scores <- matrix(0, nalg, nvm)
      rownames(scores) <- algorithms
      colnames(scores) <- validation

      for(i in 1:nalg)
        for(j in 1:nvm)
```

```
      scores[i,j] <- switch(validation[j],
    "accuracy"     = accuracy(sub.testingy,
    factor(predicted[[i]], levels=ly)),
    "sensitivity" = sensitivity(sub.testingy,
    factor(predicted[[i]], levels=ly)),
    "specificity" = specificity(sub.testingy,
    factor(predicted[[i]], levels=ly))
      )
  ## Rank aggregation to select the best model
  ## based on the performance metrics
  convScores <- convertScores(scores)
  ##-------------------------------------------
  if(nvm > 1 && nalg <= 5)
    if(weighted)
fittedModels[[k]] <- Res[[which(
algorithms == BruteAggreg(convScores$ranks,
      nalg, convScores$weights,
      distance=distance)$top.list[1])]]
  ##-------------------------------------------

  bestAlg <- unlist(sapply(fittedModels,
  FUN = function(x) attr(x, "algorithm")))
  newdata<- test.x
  predicted_sub <- list()
  Res_sub<- list()
  ## bestAlg = best.methods

  for (s in 1:nalg)
```

```
{
  Res_sub[[s]]<- switch(bestAlg[s],
                        "lasso"  =          {
    reslasso_sub <- predict(fit.lasso,
    newx = as.matrix(newdata), type = "class")
      predicted_sub[[s]]<- reslasso_sub[,1]
                        },
                        "alasso"  =          {
    resalasso_sub <- predict(fit.alasso,
    newx = as.matrix(newdata), type = "class")
    predicted_sub[[s]]<- resalasso_sub[,1]
                        },
                        "bridge"  =          {
  resenet_sub <- predict(fit.enet,
  newx = as.matrix(newdata), type = "class")
  predicted_sub[[s]]<- resenet_sub[,1]
                        },
                        "scad"  = {
  resscad_sub <- predict(fit.scad,
  X = as.matrix(newdata), type = "class")
  predicted_sub[[s]]<- resscad_sub
                        },
                        "mcp"  = {
  resmcp_sub <- predict(fit.mcp,
  X = as.matrix(newdata), type = "class")
  predicted_sub[[s]]<- resmcp_sub
                        }
  )
```

```
    }
    #store predicted classification
    mat.result[,k] = Res_sub[[1]]

    # some output
    verbose=T
    if(verbose)
      cat("Iter ", k, "\n")
  }
  majority.vote = apply(mat.result,1,function(x)
  names(table(x))[which.max(table(x))])
  acc.ens <- accuracy(test.y, majority.vote)*100
  sens.ens <- sensitivity(test.y, majority.vote)*100
  spec.ens <- specificity(test.y, majority.vote)*100

  perf <- c(acc.ens, sens.ens, spec.ens)
}
else if(method=="lasso")
{
  fit.lasso.ind = lasso(as.matrix(reduced.trainx),
  reduced.trainy)
  lasso.idx.ind = which(abs(coef(
  fit.lasso.ind)[,1])>0)[-c(1)]
  lasso.m.idx.ind = match(names(lasso.idx.ind),
  colnames(test.x))
  lasso.xb.ind = as.matrix(cbind(1,as.matrix(
  test.x[,lasso.m.idx.ind])))%*%as.matrix(coef(
  fit.lasso.ind)[c(1,lasso.idx.ind),1])
```

```
  lasso.pred.ind = exp(lasso.xb.ind)/(1+exp(lasso.xb.ind))
  lasso.pred.ind <- ifelse(lasso.pred.ind>0.5, 1, 0)
  predicted.lasso.ind <- as.character(lasso.pred.ind[,1])

  acc.lasso <- accuracy(test.y, predicted.lasso.ind)*100
  sens.lasso <- sensitivity(test.y, predicted.lasso.ind)*100
  spec.lasso <- specificity(test.y, predicted.lasso.ind)*100

  perf <- c(acc.lasso, sens.lasso, spec.lasso)
}
else if(method=="alasso")
{
  fit.alasso.ind = alasso(as.matrix(
  reduced.trainx), reduced.trainy)
  alasso.idx.ind = which(abs(coef(
  fit.alasso.ind)[,1])>0)[-c(1)]
  alasso.m.idx.ind = match(names(
  alasso.idx.ind),colnames(test.x))
  alasso.xb.ind = as.matrix(cbind(1,as.matrix(
  test.x[,alasso.m.idx.ind])))%*%as.matrix(
  coef(fit.alasso.ind)[c(1,alasso.idx.ind),1])
  alasso.pred.ind = exp(alasso.xb.ind)/(1+exp(alasso.xb.ind))
  alasso.pred.ind <- ifelse(alasso.pred.ind>0.5, 1, 0)
  predicted.alasso.ind <- as.character(alasso.pred.ind[,1])

  acc.alasso <- accuracy(test.y, predicted.alasso.ind)*100
  sens.alasso <- sensitivity(test.y, predicted.alasso.ind)*100
  spec.alasso <- specificity(test.y, predicted.alasso.ind)*100
```

```
  perf <- c(acc.alasso, sens.alasso, spec.alasso)
}
else if(method=="bridge")
{
  fit.enet.ind = enet(as.matrix(reduced.trainx),
  reduced.trainy)
  enet.idx.ind = which(abs(coef
  (fit.enet.ind)[,1])>0)[-c(1)]
  enet.m.idx.ind = match(names
  (enet.idx.ind),colnames(test.x))
  enet.xb.ind = as.matrix(cbind(1,as.matrix
  (test.x[,enet.m.idx.ind])))%*%as.matrix
  (coef(fit.enet.ind)[c(1,enet.idx.ind),1])
  enet.pred.ind = exp(enet.xb.ind)/(1+exp(enet.xb.ind))
  enet.pred.ind <- ifelse(enet.pred.ind>0.5, 1, 0)
  predicted.enet.ind <- as.character(enet.pred.ind[,1])

  acc.enet <- accuracy(test.y, predicted.enet.ind)*100
  sens.enet <- sensitivity(test.y, predicted.enet.ind)*100
  spec.enet <- specificity(test.y, predicted.enet.ind)*100

  perf <- c(acc.enet, sens.enet, spec.enet)
}
else if(method=="scad")
{
  fit.scad.ind  = scad(as.matrix(reduced.trainx),
  reduced.trainy)
```

```
  scad.idx.ind = which(abs
  (coef(fit.scad.ind))>0)[-c(1)]
  scad.m.idx.ind = match(names(scad.idx.ind),
  colnames(test.x))
  scad.xb.ind = as.matrix(
  cbind(1,as.matrix(test.x[,scad.m.idx.ind])))%*%as.matrix(
  coef(fit.scad.ind)[c(1,scad.idx.ind)])
  scad.pred.ind = exp(scad.xb.ind)/(1+exp(scad.xb.ind))
  scad.pred.ind <- ifelse(scad.pred.ind>0.5, 1, 0)
  predicted.scad.ind <- as.character(scad.pred.ind[,1])

  acc.scad <- accuracy(test.y, predicted.scad.ind)*100
  sens.scad <- sensitivity(test.y, predicted.scad.ind)*100
  spec.scad <- specificity(test.y, predicted.scad.ind)*100

  perf <- c(acc.scad, sens.scad, spec.scad)
}
else if(method=="mcp")
{
  fit.mcp.ind  = mcp(as.matrix(reduced.trainx), reduced.trainy)
  mcp.idx.ind = which(abs(coef(fit.mcp.ind))>0)[-c(1)]
  mcp.m.idx.ind = match(names(mcp.idx.ind),colnames(test.x))
  mcp.xb.ind = as.matrix(cbind(1,as.matrix(
  test.x[,mcp.m.idx.ind])))%*%as.matrix(
  coef(fit.mcp.ind)[c(1,mcp.idx.ind)])
  mcp.pred.ind = exp(mcp.xb.ind)/(1+exp(mcp.xb.ind))
  mcp.pred.ind <- ifelse(mcp.pred.ind>0.5, 1, 0)
  predicted.mcp.ind <- as.character(mcp.pred.ind[,1])
```

```
    acc.mcp <- accuracy(test.y, predicted.mcp.ind)*100
    sens.mcp <- sensitivity(test.y, predicted.mcp.ind)*100
    spec.mcp <- specificity(test.y, predicted.mcp.ind)*100

    perf <- c(acc.mcp, sens.mcp, spec.mcp)
  }
  else if(method=="svm.lin")
  {
    ## Support Vector Machine linear
    modelsvm.lin <- svm(reduced.trainx,
    reduced.trainy, kernel = "linear")
    pred_classsvm.lin <- predict(modelsvm.lin, test.x)
    ## From library caret to calculate performance metrics
    accuracy_svm.lin <- accuracy(test.y, pred_classsvm.lin)*100
sensivitiy_svm.lin <- sensitivity(test.y, pred_classsvm.lin)*100
specificity_svm.lin <- specificity(test.y, pred_classsvm.lin)*100

perf <- c(accuracy_svm.lin, sensivitiy_svm.lin,
specificity_svm.lin)
  }
  else if(method=="rf")
  {
    ## Random Forest ----------
    rfmodel <- randomForest(reduced.trainx, reduced.trainy,
    ntree= R)
    pred_classrf <- predict(rfmodel, test.x, type= "class")
```

```
  accuracy_rf <- accuracy(test.y, pred_classrf)*100
  sensitivity_rf <- sensitivity(test.y, pred_classrf)*100
  specificity_rf <- specificity(test.y, pred_classrf)*100

  perf <- c(accuracy_rf, sensitivity_rf, specificity_rf)
}
else if(method=="lr")
{
  ## logistic regression----------
  glm_lr <- cv.glmnet(as.matrix(reduced.trainx),
  reduced.trainy, family="binomial")
  pred_lr <- predict(glm_lr, newx = as.matrix(test.x),
  s = "lambda.min", type = "class")

  accuracy_lr <- accuracy(test.y, pred_lr)*100
  sensitivity_lr <- sensitivity(test.y, pred_lr)*100
  specificity_lr <- specificity(test.y, pred_lr)*100

  perf <- c(accuracy_lr, sensitivity_lr, specificity_lr)
}
else if(method=="adab")
{
  ## Ada Boost ----------------------------------------
  model_adab <- boosting(reduced.trainy~., data = dat.train,
  boos=TRUE, mfinal=R)
  pred_adab <- predict(model_adab, as.data.frame(test.x))

  accuracy_adab <- accuracy(test.y, pred_adab$class)*100
```

```
    sensitivity_adab <- sensitivity(test.y, pred_adab$class)*100
    specificity_adab <- specificity(test.y, pred_adab$class)*100

    perf <- c(accuracy_adab, sensitivity_adab, specificity_adab)
  }
  return(perf)
}

## Function to calculate performance metrics
perf.metrics <- function(results)
{
  ens.acc <- results[,1]/100; lasso.acc <- results[,4]/100;
  alasso.acc <- results[,7]/100;
  enet.acc <- results[,10]/100;scad.acc <- results[,13]/100;
  mcp.acc <- results[,16]/100; svm.lin.acc <- results[,19]/100;
  rf.acc <- results[,22]/100;
  lr.acc <- results[,25]/100; adab.acc <- results[,28]/100;

  model <- data.frame(Methods = c(rep("ERRM", reps),
  rep("LASSO", reps), rep("ALASSO", reps),
  rep("ENET", reps), rep("SCAD", reps), rep("MCP", reps),
  rep("SVM", reps), rep("RF", reps),rep("LR", reps),
  rep("AB", reps)),
  Accuracy = c(ens.acc, lasso.acc, alasso.acc, enet.acc,
  scad.acc, mcp.acc, svm.lin.acc, rf.acc, lr.acc, adab.acc))


  ensemble.accuracy <- c(mean(results[,1]), sd(results[,1]))
```

```
ensemble.sensitivity <- c(mean(results[,2]), sd(results[,2]))
ensemble.specificity <- c(mean(results[,3]), sd(results[,3]))
lasso.accuracy <- c(mean(results[,4]), sd(results[,4]))
lasso.sensitivity <- c(mean(results[,5]), sd(results[,5]))
lasso.specificity <- c(mean(results[,6]), sd(results[,6]))
alasso.accuracy <- c(mean(results[,7]), sd(results[,7]))
alasso.sensitivity <- c(mean(results[,8]), sd(results[,8]))
alasso.specificity <- c(mean(results[,9]), sd(results[,9]))
enet.accuracy <- c(mean(results[,10]), sd(results[,10]))
enet.sensitivity <- c(mean(results[,11]), sd(results[,11]))
enet.specificity <- c(mean(results[,12]), sd(results[,12]))
scad.accuracy <- c(mean(results[,13]), sd(results[,13]))
scad.sensitivity <- c(mean(results[,14]), sd(results[,14]))
scad.specificity <- c(mean(results[,15]), sd(results[,15]))
mcp.accuracy <- c(mean(results[,16]), sd(results[,16]))
mcp.sensitivity <- c(mean(results[,17]), sd(results[,17]))
mcp.specificity <- c(mean(results[,18]), sd(results[,18]))
svm.lin.accuracy <- c(mean(results[,19]), sd(results[,19]))
svm.lin.sensitivity <- c(mean(results[,20]), sd(results[,20]))
svm.lin.specificity <- c(mean(results[,21]), sd(results[,21]))
rf.accuracy <- c(mean(results[,22]), sd(results[,22]))
rf.sensitivity <- c(mean(results[,23]), sd(results[,23]))
rf.specificity <- c(mean(results[,24]), sd(results[,24]))
lr.accuracy <- c(mean(results[,25]), sd(results[,25]))
lr.sensitivity <- c(mean(results[,26]), sd(results[,26]))
lr.specificity <- c(mean(results[,27]), sd(results[,27]))
adab.accuracy <- c(mean(results[,28]), sd(results[,28]))
adab.sensitivity <- c(mean(results[,29]), sd(results[,29]))
```

```
adab.specificity <- c(mean(results[,30]), sd(results[,30]))


ensemble.performance <- list(ensemble.accuracy,
ensemble.sensitivity, ensemble.specificity)
names(ensemble.performance) <- c("ensemble accuracy",
"ensemble sensitivity","ensemble specificity")
lasso.performance <- list(lasso.accuracy,
lasso.sensitivity, lasso.specificity)
names(lasso.performance) <- c("lasso accuracy",
"lasso sensitivity","lasso specificity")
alasso.performance <- list(alasso.accuracy,
alasso.sensitivity, alasso.specificity)
names(alasso.performance) <- c("alasso accuracy",
"alasso sensitivity","alasso specificity")
enet.performance <- list(enet.accuracy, enet.sensitivity,
enet.specificity)
names(enet.performance) <- c("enet accuracy",
"enet sensitivity","enet specificity")
scad.performance <- list(scad.accuracy,
scad.sensitivity, scad.specificity)
names(scad.performance) <- c("scad accuracy",
"scad sensitivity","scad specificity")
mcp.performance <- list(mcp.accuracy,
mcp.sensitivity, mcp.specificity)
names(mcp.performance) <- c("mcp accuracy",
"mcp sensitivity","mcp specificity")
svm.lin.performance <- list(svm.lin.accuracy,
svm.lin.sensitivity, svm.lin.specificity)
```

```
  names(svm.lin.performance) <- c("svm.lin accuracy",
  "svm.lin sensitivity","svm.lin specificity")
  rf.performance <- list(rf.accuracy, rf.sensitivity,
  rf.specificity)
  names(rf.performance) <- c("rf accuracy", "rf sensitivity",
  "rf specificity")
  lr.performance <- list(lr.accuracy, lr.sensitivity,
  lr.specificity)
  names(lr.performance) <- c("lr accuracy", "lr sensitivity",
  "lr specificity")
  adab.performance <- list(adab.accuracy, adab.sensitivity,
  adab.specificity)
  names(adab.performance) <- c("adab accuracy",
  "adab sensitivity","adab specificity")

  return(list(ensemble.performance, lasso.performance,
  alasso.performance, enet.performance, scad.performance,
  mcp.performance, svm.lin.performance, rf.performance,
  lr.performance, adab.performance, model))
}
```

Curriculum Vitae

Abhijeet R Patil was born on June 29, 1990. The second son of Revansiddappa S Patil and Jyothi R Patil, he entered Sai Vidya Institute of Technology (SVIT), Bangalore, India, in the fall of 2008, he obtained his bachelor's degree in Computer Science and Engineering in the summer of 2012. In the fall of 2012, he entered R.V. College of Engineering, Bangalore, India. While pursuing a master's degree in Bioinformatics, he worked as a Teaching Assistant in the Biotechnology department. In 2015 he joined Sharnbasva University (previously called Appa Institute of Engineering and Technology), he worked as Assistant Professor where he taught various computer science and engineering courses over two years. He has been granted sabbatical by the institution from August 2017 till August 2021 for pursuing his Ph.D. studies in the USA. Abhijeet got accepted in the Computation Science program at The University of Texas at El Paso to pursue his Doctoral degree in fall 2017 and currently resides in El Paso with his wife.

Email address: arpatil@miners.utep.edu